

## Contents

<b>S.No</b>	<b>Topic</b>	<b>Page. No.</b>
1	Cover Page	
2	Syllabus copy	
3	Vision of the Department	
4	Mission of the Department	
5	PEOs and POs	
6	Course objectives and outcomes	
7	Course mapping with POs	
8	Brief notes on the importance of the course and how it fits into the curriculum	
9	Prerequisites if any	
10	Instructional Learning Outcomes	
11	Class Time Table	
12	Individual Time Table	
13	Lecture schedule with methodology being used/adopted	
14	Detailed notes	
15	Additional topics	
16	University Question papers of previous years	
17	Question Bank	
18	Assignment Questions	
19	Unit wise Quiz Questions and long answer questions	
20	Tutorial problems	
21	Known gaps ,if any and inclusion of the same in lecture schedule	
22	Discussion topics , if any	

<b>23</b>	References, Journals, websites and E-links if any	
<b>24</b>	Quality Measurement Sheets	
<b>A</b>	Course End Survey	
<b>B</b>	Teaching Evaluation	
<b>25</b>	Student List	
<b>26</b>	Group-Wise students list for discussion topic	

**Course coordinator**

**Program Coordinator**

**HOD**

# Geethanjali College of Engineering and Technology

Cheeryal (V), Keesara (M), Ranga Reddy District – 501 301 (T.S)

## JAVA PROGRAMMING

Course File



Geethanjali

DEPARTMENT OF  
COMPUTER SCIENCE & ENGINEERING

Lab In charge

HOD-CSE

# Geethanjali College of Engineering and Technology

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Name of the Subject/Lab Course): Java Programming

(JNTU CODE: A40503 )

Programme: UG

Branch: CSE

Version No: 1

Year: II A & B ,C&D

Document Number :GCET/CSE/

Semester: I

No. of Pages:

Classification status (Unrestricted/Restricted) : Unrestricted

Distribution List: Unrestricted

Prepared by :

1) Name : A.Sree Lakshmi

2) Sign :

3) Design : ASSOCIATE PROFESSOR

4) Date :

1) Name : N. Swapna

2) Sign :

3) Design : ASSOCIATE PROFESSOR

4) Date :

Verified by :

\*For Q.C only

1) Name :

1)Name :

2) Sign :

2) Sign :

3) Design :

3) Design :

4) Date :

4) Date :

Approved by (HOD) :

1) Name :

2) Sign :

3) Date :

## 2) Syllabus copy

### Syllabus

#### UNIT-I

##### **OOP concepts:**

- Data abstraction , encapsulation , inheritance, benefits of inheritance , polymorphism, classes and objects , procedural and Object oriented programming paradigms

##### **Java Programming**

-History of Java, comments, datatypes, variables, constants, , scope and life time of variables,operators,operator hierarchy, expressions, type conversion and casting, enumerated types,control flow block scope, conditional statements, loops break and continue statements.simple java program ,arrays,console input and output, formatting output, constructors, methods, parameter passing, static fields and methods, access control, this keyword, overloading methods and constructors recursion,garbage collection,building strings, exploring string class

#### UNIT-II

**Inheritance** –Definition ,hierarchies, super and subclasses , Member access rules, super keyword,preventing inheritance : final classes and methods , the Object class and its methods.

**Polymorphism**- Dynamic binding, method overriding, abstract classes and methods .

**Interfaces** : Interfaces VS Abstrct classes,defining an interface , implementing interfaces, accessing implementations through interface references, extending interface.

**Inner classes:** Uses of inner classes, local inner classes, anonymous inner classes, static inner classes, examples.

**Packages:** Definition, Creating and Accessing a package, understanding CLASSPATH, importing packages.

#### UNIT-III

**Exception handling** – Dealing with errors, benefits of exception handling, the classification of exceptions- exception hierarchy, checked exceptions and unchecked exceptions, usage of try, catch, throw, throws and finally, rethrowing exceptions, exception specification,built in exceptions, creating own exception sub classes.

**Multi-Threading:-** Differences between multiple processes and multiple threads, thread states, creating threads, interrupting threads, thread priorities, synchronizing threads, inter thread communication, producer consumer pattern.

#### **UNIT-IV**

**Collection Framework in java:** Introduction to java Collections, overview of java collection framework, Generics, commonly used collection classes- ArrayList, Vector, Hash table, Stack, Enumeration, Iterator, String tokenizer, Random, Scanner, Calendar and Properties

**Files: streams** – byte streams, character streams, text input/ Output binary input/ output  
Random access file operations, file management using file class.

**Connecting to Database-JDBC** type 1 to 4 drivers , connecting to a data base , querying a data base and processing the results, updating data with JDBC.

#### **UNIT –V**

**GUI Programming with java-**The AWT class hierarchy, Introduction to Swing, Swing VS AWT, Hierarchy for Swing components, containers-JFrame, JApplet, JDialog, JPanel, Overview of some swing components-Jbutton, JLabel, JTextField, JTextarea, simple Swing Applications, LayoutManagement- Layout Manager types- border , grid and flow

**Event handling:** Events, event sources , event classes, event Listeners, Relationship between event sources and Listeners Delegation event model, Examples: handling a button click, handling mouse events, Adapter classes.

Applets – Inheritance hierarchy for applets, differences between applets and applications ,life cycle of an applet, passing parameters , applet security issues.

#### **TEXT BOOKS**

. .1. Java fundamentals- A comprehensive Introduction Herbert Schildt and Dale Skrien, TMH

#### **REFERENCE BOOKS:**

1. **Java for Programmers, P.J. Dietel and H.M Deitel Pearson education.**
2. Object Oriented Programming through Java. P.Radha Krishna. Universities Press.
3. Thinking in Java Bruce Eckel, Pearson Education.
4. Programming in Java , S. Malhotra and S. Choudhary , Oxford University Press.

### **3. Vision of the Department**

To produce globally competent and socially responsible computer science engineers contributing to the advancement of engineering and technology which involves creativity and innovation by providing excellent learning environment with world class facilities.

### **4. Mission of the Department**

1. To be a center of excellence in instruction, innovation in research and scholarship, and service to the stake holders, the profession, and the public.
2. To prepare graduates to enter a rapidly changing field as a competent computer science engineer.
3. To prepare graduate capable in all phases of software development, possess a firm understanding of hardware technologies, have the strong mathematical background necessary for scientific computing, and be sufficiently well versed in general theory to allow growth within the discipline as it advances.
4. To prepare graduates to assume leadership roles by possessing good communication skills, the ability to work effectively as team members, and an appreciation for their social and ethical responsibility in a global setting.

### **5. PROGRAM EDUCATIONAL OBJECTIVES (PEOs) OF C.S.E. DEPARTMENT**

1. To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in industry and / or to pursue postgraduate studies with an appreciation for lifelong learning.
2. To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

3. To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting-edge technologies of multi-disciplinary nature for societal development.

#### **6.PROGRAM OUTCOMES (CSE)**

1. An ability to apply knowledge of mathematics, science and engineering to develop and analyze computing systems.
2. an ability to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints.
3. An ability to perform experiments to analyze and interpret data for different applications.
4. An ability to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space.
5. An ability to use current techniques, skills and modern engineering tools necessary to practice as a CSE professional.
6. An ability to recognize the importance of professional, ethical, legal, security and social issues and addressing these issues as a professional.
7. An ability to analyze the local and global impact of systems /processes /applications /technologies on individuals, organizations, society and environment.
8. An ability to function in multidisciplinary teams.
9. An ability to communicate effectively with a range of audiences.
10. Demonstrate knowledge and understanding of the engineering, management and economic principles and apply them to manage projects as a member and leader in a team.
11. A recognition of the need for and an ability to engage in life-long learning and continuing professional development
12. Knowledge of contemporary issues.
13. An ability to apply design and development principles in producing software systems of varying complexity using various project management tools.
14. An ability to identify, formulate and solve innovative engineering problems.

## **7.Course objectives and outcomes**

### **Course Objectives:**

- 1) To understand Object Oriented Programming concepts and apply them in problem solving.
- 2) To learn the basics of java concepts and GUI based programming.

### **Course Outcomes**

Upon completion of this course, students would be able to:

**A40503.1.** Understand the concept of OOP as well as the purpose and usage principles of inheritance, polymorphism, encapsulation and method overloading.

**A40503.2.** Identify classes, objects, members of a class and the relationships among them needed for a specific problem.

**A40503.3.** Create Java application programs using sound OOP practices (e.g., interfaces and APIs) and proper program structuring (e.g., by using access control identifies, automatic documentation through comments, error exception handling)

**A40503.4.** Develop programs using the Java Collection API as well as the Java standard class library.

**A40503.5.** Develop the skills to apply java programming in problem solving.

**A40503.6.** Should get the ability to extend his/her knowledge of java programming further on his her own.

## **8 Brief notes on the importance of the course and how it fits into the curriculum**

This course introduces students to object oriented programming and design. Students will be exposed to the principles of the object oriented programming paradigm specifically including abstraction, encapsulation, inheritance and polymorphism. They are equipped to use an object oriented programming language Java, and associated class libraries, to develop object oriented programs with a clear understanding of Java features. The course helps the students to design, develop, test, and debug programs using object oriented principles, GUI design with Applets and Swings, JDBC –ODBC connections in conjuncture with an integrated development environment Eclipse. They will be able to describe and explain the factors that contribute to a good object oriented solution.

The course enables a student to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints, to perform experiments to analyze and interpret data for different applications, and to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space. The student can use current techniques, skills and tools necessary to practice as a CSE professional and continuing professional development. The student gains knowledge of subject to apply design and development principles in producing software systems of varying complexity

## **9 Prerequisites if any**

- i) Knowledge in programming i.e., C language

## **10 Instructional Learning Outcomes**

### **Unit-I:**

Students gain ability to

1. Explain and apply object oriented concepts
2. Differentiate Procedural and object oriented paradigms
3. Explain features of Java programming language
4. Write simple java stand alone applications

### **Unit-II**

1. Apply the Concepts of inheritance, Polymorphism and Interfaces and Packages
2. Design abstract classes and interfaces and analyze when to use them.
2. Create and access a package.

### **Unit-III**

1. Handle Exceptions and create own exception sub classes.
2. Multithread programming with an ability to synchronize threads.

### **Unit- IV**

1. Use the Collection Frame work in java.

2. File management with a clear understanding of different kinds of input output streams
3. Connecting and querying a database using JDBC

### Unit-V

1. Design GUI screens using AWT and Swings
2. Handle different types of events
3. Design Applets

## 11 Course mapping with POs

Course Code and Title	POS													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
<b>A40503-</b> Java Programming														
<b>A40503.1</b> Understand the concept of OOP as well as the purpose and usage principles of inheritance, polymorphism, encapsulation and method overloading	2	2	2	2	1		1	1		1			2	2
<b>A40503.2</b> Identify classes, objects, members of a class and the relationships among them needed for a specific problem.	2	2	2	2	1		1	1		1			2	2
<b>A40503.3.</b> Create Java application programs using sound OOP practices (e.g., interfaces and APIs) and proper	2	2	2	2	1		1	1		1			2	2

program structuring (e.g., by using access control identifies, automatic documentation through comments, error exception handling)														
<b>A40503.4.</b> Develop programs using the Java Collection API as well as the Java standard class library.	2	2	2	2	1		1	1		1			2	2
<b>A40503.5.</b> Develop the skills to apply java programming in problem solving.	2	2	2	2	1		1	1		1			2	2
<b>A40503.6.</b> Should get the ability to extend his/her knowledge of java programming further on his her own.	2	2	2	2	1		1	1		1	1		2	2

## 12 Class Time Table.

## 13 Lecture schedule with methodology being used/adopted

### Section A

SNO	LECTURE NO	TOPIC NAME	Methodology OHP/Board/LCD	
<b>UNIT I</b>				
1	1	Introduction to subject: Objectives , Outcomes	Board	
2	1	Object Oriented Programming Concepts.	Board	
3	1	Procedural and Object Oriented Paradigms	Board	
4	1	History of JAVA	Board	
5	1	Data Types, Variables ,constants ,Scope and lifetime of variables	Board	
4	1	Creating a simple java program, compiling and running,	Board	

5	1	Operators, expressions, Type conversion and casting	Board	
6	1	Control Statements	Board	
7	1	Arrays, console input and output	Board	
8	1	classes and objects, Concepts of classes, constructors and methods	Board	
9	1	Access control, this keyword, garbage collection	Board	
10	1	Overloading methods and constructors	Board	
11	1	Recursion, String handling	Board	
<b>Total</b>	<b>13</b>		Board	
<b>UNIT II</b>				
12	1	Inheritance definition, single inheritance, benefits of inheritance	Board	
13	1	Super classes and subclass Member access rules	Board	
14	1	Preventing inheritance: final class and methods	Board	
15	1	Base class object and its methods	Board	
16	1	Polymorphism: dynamic Binding	Board	
17	1	Method overriding	Board	
18	1	Abstract classes and methods	Board	
19	1	Defining an interface, implementing interface	Board	
20	1	Implements and extends keywords	Board	
21	1	An application using an interfaces and uses of interfaces	Board	
22	1	Differences between abstract classes and interfaces	Board	
23	1	Inner classes (local and Anonymous and static)	Board	
24	1	examples	Board	
25	1	Defining, Creating and Accessing a Package, Types of packages	Board	
26	1	Understanding CLASSPATH, importing packages	Board	
Total	15			
<b>UNIT III</b>				
27	1	Dealing with errors and benefits of Exception Handling	Board	
28	1	Concepts of Exception handling, Exception hierarchy	Board	
29	1	Types of exceptions, usage of try, catch, throw, throws, finally keywords	Board	
30	1	Built in Exceptions, Creating own Exception sub classes	Board	
31	1	Differences between multiple Processes and Multiple threads	Board	
32	1	Thread life cycle, creating multiple threads by using thread	Board	

		class		
33	1	Creating multiple threads by using Runnable interface	Board	
34	1	Interrupting threads and thread priorities	Board	
35	1	Synchronization	Board	
36	1	Inter thread communication	Board	
Total	10			
<b>UNIT IV</b>				
37	1	Introduction and Overview of Collection framework	Board	
38	1	Generics	Board	
39	1	Commonly used Classes- ArrayList, Vector	Board	
40	1	Hashtable, stack	Board	
41	1	Enumeration and iterator	Board	
42	1	String tokenizer	Board	
43	1	Random, scanner, calendar and properties	Board	
44	1	Files- streams	Board	
45	1	Text input/output	Board	
46	1	binary input/output	Board	
47	1	Random access file operations, File management using file class	Board	
48	1	Connecting to database : drivers – 4 types	Board	
49	1	Connection and querying the database, Updating data with JDBC	Board	
Total	13			
<b>UNIT V</b>				
50	1	Advantages of GUI over CUI, The AWT class hierarchy	Board	
51	1	Introduction to swing and Swing VS AWT	Board	
52	1	Hierarchy for swing components	Board	
53	1	containers	Board	
54	1	Jbutton,jtextfield,jtextarea with example	Board	
55	1	Layout managers(border, grid and flow)	Board	
56	1	Event handling	Board	
57	1	Events, sources, classes, listeners	Board	
58	1	Relationship between event sources and listeners	Board	
59	1	Delegation Event model	Board	
60	1	Handling button click	Board	
61	1	Handling mouse events	Board	

62	1	Concepts of Applets, differences between applets and, Applications	Board	
63	1	Life cycle of an applet	Board	
64	1	Types of applets, creating applets, passing parameters to applets	Board	
65	1	Applet security	Board	
Total	16			

### **Section B**

<b>SNO</b>	<b>LECTURE NO</b>	<b>TOPIC NAME</b>	<b>Methodology</b> OHP/Board/LCD	
<b>UNIT I</b>				
1	1	Introduction to subject: Objectives , Outcomes	Board	
2	1	Object Oriented Programming Concepts.	Board	
3	1	Procedural and Object Oriented Paradigms	Board	
4	1	History of JAVA	Board	
5	1	Data Types, Variables ,constants ,Scope and lifetime of variables	Board	
4	1	Creating a simple java program, compiling and running,	Board	
5	1	Operators, expressions,Type conversion and casting	Board	
6	1	Control Statements	Board	
7	1	Arrays, console input and output	Board	
8	1	classes and objects, Concepts of classes, constructors and methods	Board	
9	1	Access control, this keyword, garbage collection	Board	
10	1	Overloading methods and constructors	Board	
11	1	Recursion, String handling	Board	
<b>Total</b>	<b>13</b>		Board	
<b>UNIT II</b>				
12	1	Inheritance definition, single inheritance, benefits of inheritance	Board	
13	1	Super classes and subclass Member access rules	Board	
14	1	Preventing inheritance:final class and methods	Board	
15	1	Base class object and its methods	Board	
16	1	Polymorphism: dynamic Binding	Board	
17	1	Method overriding	Board	
18	1	Abstract classes and methods	Board	

19	1	Defining an interface, implementing interface	Board	
20	1	Implements and extends keywords	Board	
21	1	An application using an interfaces and uses of interfaces	Board	
22	1	Differences between abstract classes and interfaces	Board	
23	1	Inner classes(local and Anonymous and static)	Board	
24	1	examples	Board	
25	1	Defining, Creating and Accessing a Package, Types of packages	Board	
26	1	Understanding CLASSPATH, importing packages	Board	
Total	15			
<b>UNIT III</b>				
27	1	Dealing with errors and benefits of Exception Handling	Board	
28	1	Concepts of Exception handling, Exception hierarchy	Board	
29	1	Types of exceptions, usage of try, catch, throw, throws, finally keywords	Board	
30	1	Built in Exceptions, Creating own Exception sub classes	Board	
31	1	Differences between multiple Processes and Multiple threads	Board	
32	1	Thread life cycle, creating multiple threads by using thread class	Board	
33	1	Creating multiple threads by using Runnable interface	Board	
34	1	Interrupting threads and thread priorities	Board	
35	1	Synchronization	Board	
36	1	Inter thread communication	Board	
Total	10			
<b>UNIT IV</b>				
37	1	Introduction and Overview of Collection framework	Board	
38	1	Generics	Board	
39	1	Commonly used Classes- ArrayList, Vector	Board	
40	1	Hashtable, stack	Board	
41	1	Enumeration and iterator	Board	
42	1	String tokenizer	Board	
43	1	Random, scanner, calendar and properties	Board	
44	1	Files- streams	Board	
45	1	Text input/output	Board	
46	1	binary input/output	Board	

47	1	Random access file operations, File management using file class	Board	
48	1	Connecting to database : drivers – 4 types	Board	
49	1	Connection and querying the database, Updating data with JDBC	Board	
Total	13			
<b>UNIT V</b>				
50	1	Advantages of GUI over CUI, The AWT class hierarchy	Board	
51	1	Introduction to swing and Swing VS AWT	Board	
52	1	Hierarchy for swing components	Board	
53	1	containers	Board	
54	1	Jbutton,jtextfield,jtextarea with example	Board	
55	1	Layout managers(border, grid and flow)	Board	
56	1	Event handling	Board	
57	1	Events, sources, classes, listeners	Board	
58	1	Relationship between event sources and listeners	Board	
59	1	Delegation Event model	Board	
60	1	Handling button click	Board	
61	1	Handling mouse events	Board	
62	1	Concepts of Applets, differences between applets and, Applications	Board	
63	1	Life cycle of an applet	Board	
64	1	Types of applets, creating applets, passing parameters to applets	Board	
65	1	Applet security	Board	
Total	16			

### **Section C**

SNO	LECTURE NO	TOPIC NAME	Methodology OHP/Board/LCD	
<b>UNIT I</b>				
1	1	Introduction to subject: Objectives , Outcomes	Board	
2	1	Object Oriented Programming Concepts.	Board	
3	1	Procedural and Object Oriented Paradigms	Board	
4	1	History of JAVA	Board	
5	1	Data Types, Variables ,constants ,Scope and lifetime of variables	Board	

4	1	Creating a simple java program, compiling and running,	Board	
5	1	Operators, expressions, Type conversion and casting	Board	
6	1	Control Statements	Board	
7	1	Arrays, console input and output	Board	
8	1	classes and objects, Concepts of classes, constructors and methods	Board	
9	1	Access control, this keyword, garbage collection	Board	
10	1	Overloading methods and constructors	Board	
11	1	Recursion, String handling	Board	
<b>Total</b>	<b>13</b>		Board	
<b>UNIT II</b>				
12	1	Inheritance definition, single inheritance, benefits of inheritance	Board	
13	1	Super classes and subclass Member access rules	Board	
14	1	Preventing inheritance: final class and methods	Board	
15	1	Base class object and its methods	Board	
16	1	Polymorphism: dynamic Binding	Board	
17	1	Method overriding	Board	
18	1	Abstract classes and methods	Board	
19	1	Defining an interface, implementing interface	Board	
20	1	Implements and extends keywords	Board	
21	1	An application using an interfaces and uses of interfaces	Board	
22	1	Differences between abstract classes and interfaces	Board	
23	1	Inner classes(local and Anonymous and static)	Board	
24	1	examples	Board	
25	1	Defining, Creating and Accessing a Package, Types of packages	Board	
26	1	Understanding CLASSPATH, importing packages	Board	
Total	15			
<b>UNIT III</b>				
27	1	Dealing with errors and benefits of Exception Handling	Board	
28	1	Concepts of Exception handling, Exception hierarchy	Board	
29	1	Types of exceptions, usage of try, catch, throw, throws, finally keywords	Board	
30	1	Built in Exceptions, Creating own Exception sub classes	Board	
31	1	Differences between multiple Processes and Multiple threads	Board	

32	1	Thread life cycle, creating multiple threads by using thread class	Board	
33	1	Creating multiple threads by using Runnable interface	Board	
34	1	Interrupting threads and thread priorities	Board	
35	1	Synchronization	Board	
36	1	Inter thread communication	Board	
Total	10			
<b>UNIT IV</b>				
37	1	Introduction and Overview of Collection framework	Board	
38	1	Generics	Board	
39	1	Commonly used Classes- ArrayList, Vector	Board	
40	1	Hashtable, stack	Board	
41	1	Enumeration and iterator	Board	
42	1	String tokenizer	Board	
43	1	Random, scanner, calendar and properties	Board	
44	1	Files- streams	Board	
45	1	Text input/output	Board	
46	1	binary input/output	Board	
47	1	Random access file operations, File management using file class	Board	
48	1	Connecting to database : drivers – 4 types	Board	
49	1	Connection and querying the database, Updating data with JDBC	Board	
Total	13			
<b>UNIT V</b>				
50	1	Advantages of GUI over CUI, The AWT class hierarchy	Board	
51	1	Introduction to swing and Swing VS AWT	Board	
52	1	Hierarchy for swing components	Board	
53	1	containers	Board	
54	1	Jbutton,jtextfield,jtextarea with example	Board	
55	1	Layout managers(border, grid and flow)	Board	
56	1	Event handling	Board	
57	1	Events, sources, classes, listeners	Board	
58	1	Relationship between event sources and listeners	Board	
59	1	Delegation Event model	Board	
60	1	Handling button click	Board	
61	1	Handling mouse events	Board	

62	1	Concepts of Applets, differences between applets and, Applications	Board	
63	1	Life cycle of an applet	Board	
64	1	Types of applets, creating applets, passing parameters to applets	Board	
65	1	Applet security	Board	
Total	16			
SNO	LECTURE NO	TOPIC NAME	Methodology	
<b>UNIT I</b>				
1	1	Introduction to subject: Objectives , Outcomes	Board	
2	1	Object Oriented Programming Concepts.	Board	
3	1	Procedural and Object Oriented Paradigms	Board	
4	1	History of JAVA	Board	
5	1	Data Types, Variables ,constants ,Scope and lifetime of variables	Board	
4	1	Creating a simple java program, compiling and running,	Board	
5	1	Operators, expressions,Type conversion and casting	Board	
6	1	Control Statements	Board	
7	1	Arrays, console input and output	Board	
8	1	classes and objects, Concepts of classes, constructors and methods	Board	
9	1	Access control, this keyword, garbage collection	Board	
10	1	Overloading methods and constructors	Board	
11	1	Recursion, String handling	Board	
<b>Total</b>	<b>13</b>		Board	
<b>UNIT II</b>				
12	1	Inheritance definition, single inheritance, benefits of inheritance	Board	
13	1	Super classes and subclass Member access rules	Board	
14	1	Preventing inheritance:final class and methods	Board	
15	1	Base class object and its methods	Board	
16	1	Polymorphism: dynamic Binding	Board	
17	1	Method overriding	Board	
18	1	Abstract classes and methods	Board	
19	1	Defining an interface, implementing interface	Board	
20	1	Implements and extends keywords	Board	

21	1	An application using an interfaces and uses of interfaces	Board	
22	1	Differences between abstract classes and interfaces	Board	
23	1	Inner classes(local and Anonymous and static)	Board	
24	1	examples	Board	
25	1	Defining, Creating and Accessing a Package, Types of packages	Board	
26	1	Understanding CLASSPATH, importing packages	Board	
Total	15			
<b>UNIT III</b>				
27	1	Dealing with errors and benefits of Exception Handling	Board	
28	1	Concepts of Exception handling, Exception hierarchy	Board	
29	1	Types of exceptions, usage of try, catch, throw, throws, finally keywords	Board	
30	1	Built in Exceptions, Creating own Exception sub classes	Board	
31	1	Differences between multiple Processes and Multiple threads	Board	
32	1	Thread life cycle, creating multiple threads by using thread class	Board	
33	1	Creating multiple threads by using Runnable interface	Board	
34	1	Interrupting threads and thread priorities	Board	
35	1	Synchronization	Board	
36	1	Inter thread communication	Board	
Total	10			
<b>UNIT IV</b>				
37	1	Introduction and Overview of Collection framework	Board	
38	1	Generics	Board	
39	1	Commonly used Classes- ArrayList, Vector	Board	
40	1	Hashtable, stack	Board	
41	1	Enumeration and iterator	Board	
42	1	String tokenizer	Board	
43	1	Random, scanner, calendar and properties	Board	
44	1	Files- streams	Board	
45	1	Text input/output	Board	
46	1	binary input/output	Board	
47	1	Random access file operations, File management using file class	Board	
48	1	Connecting to database : drivers – 4 types	Board	

49	1	Connection and querying the database, Updating data with JDBC	Board	
Total	13			
<b>UNIT V</b>				
50	1	Advantages of GUI over CUI, The AWT class hierarchy	Board	
51	1	Introduction to swing and Swing VS AWT	Board	
52	1	Hierarchy for swing components	Board	
53	1	containers	Board	
54	1	Jbutton,jtextfield,jtextarea with example	Board	
55	1	Layout managers(border, grid and flow)	Board	
56	1	Event handling	Board	
57	1	Events, sources, classes, listeners	Board	
58	1	Relationship between event sources and listeners	Board	
59	1	Delegation Event model	Board	
60	1	Handling button click	Board	
61	1	Handling mouse events	Board	
62	1	Concepts of Applets, differences between applets and, Applications	Board	
63	1	Life cycle of an applet	Board	
64	1	Types of applets, creating applets, passing parameters to applets	Board	
65	1	Applet security	Board	
Total	16			

## 14 Detailed notes

Java is a **programming language** and a **platform**.

Java is a high level, robust, secured and object-oriented programming language.

**Platform:** Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

---

## Java Example

Let's have a quick look at java programming example. A detailed description of hello java example is given in next page.

```
1. class Simple{
2.     public static void main(String args[]){
3.         System.out.println("Hello Java");
4.     }
5. }
```

### **Uses of java**

According to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
  2. Web Applications such as irctc.co.in, javatpoint.com etc.
  3. Enterprise Applications such as banking applications.
  4. Mobile
  5. Embedded System
  6. Smart Card
  7. Robotics
  8. Games etc.
- 

## **Types of Java Applications**

There are mainly 4 types of applications that can be created using java programming:

### **1) Standalone Application**

It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

### **2) Web Application**

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

### 3) Enterprise Application

An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

### 4) Mobile Application

An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

## History of Java

1. Brief history of Java
2. Java Version History

**Java history** is interesting to know. The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

For the green team members, it was an advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

- 1) **James Gosling, Mike Sheridan, and Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.
- 2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.
- 3) Firstly, it was called "**Greentalk**" by James Gosling and file extension was .gt.
- 4) After that, it was called **Oak** and was developed as a part of the Green project.

### Why Oak name for java language?

5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.

6) In 1995, Oak was renamed as "**Java**" because it was already a trademark by Oak Technologies.

### Why Java name for java language?

7) **Why they choosed java name for java language?** The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

According to James Gosling "Java was one of the top choices along with **Silk**". Since java was so unique, most of the team members preferred java.

8) Java is an island of Indonesia where first coffee was produced (called java coffee).

9) Notice that Java is just a name not an acronym.

10) Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

## **1.2 Basic Concept of OOP(Object-Oriented Programming):**

There are some basic concepts of object oriented programming as follows:

1. Object
2. Class
3. Data abstraction
4. Data encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic binding

### **1. Object**

Objects are important runtime entities in object oriented method. They may characterize a location, a bank account, and a table of data or any entry that the program must handle.

Each object holds data and code to operate the data. Object can interact without having to identify the details of each other's data or code. It is sufficient to identify the type of message received and the type of reply returned by the objects.

example of object is CAR

Object: CAR
DATA
Colour
Cost
METHODS
LockIt()
DriveIt()

Fig.1.2 Representation of object —CAR

Fig.1.1 and Fig.1.2 shows actual representation of object.

## 2. Classes

A class is a set of objects with similar properties (attributes), common behaviour (operations), and common link to other objects. The complete set of data and code of an object can be made a user defined data type with the help of class.

The objects are variable of type class. A class is a collection of objects of similar type. Classes are user defined data types and work like the built in type of the programming language. Once the class has been defined, we can make any number of objects belonging to that class. Each object is related with the data of type class with which they are formed.

As we learned that, the classification of objects into various classes is based on its properties (States) and behaviour (methods). Classes are used to distinguish are type of object from another. The important thing about the class is to identify the properties and procedures and applicability to its instances.

In above example, we will create an objects MH-01 1234 belonging to the class car. The objects develop their distinctiveness from the difference in their attribute value and relationships to other objects.

### **3. Data Abstraction**

Data abstraction refers to the act of representing important description without including the background details or explanations.

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, cost and functions operate on these attributes. They summarize all the important properties of the objects that are to be created.

Classes use the concepts of data abstraction and it is called as Abstract Data Type (ADT).

### **4. Data Encapsulation**

Data Encapsulation means wrapping of data and functions into a single unit (i.e. class). It is most useful feature of class. The data is not easy to get to the outside world and only those functions which are enclosed in the class can access it.

These functions provide the boundary between Object's data and program. This insulation of data from direct access by the program is called as **Data hiding**.

### **5. Inheritance**

Inheritance is the process by which objects of one class can get the properties of objects of another class. Inheritance means one class of objects inherits the data and behaviours from another class. Inheritance maintains the hierarchical classification in which a class inherits from its parents.

Inheritance provides the important feature of OOP that is reusability. That means we can include additional characteristics to an existing class without modification. This is possible deriving a new class from existing one.

In other words, it is property of object-oriented systems that allow objects to be built from other objects. Inheritance allows openly taking help of the commonality of objects when constructing new classes. Inheritance is a relationship between classes where one class is the parent class of another (derived) class. The derived class holds the properties and behaviour of base class in addition to the properties and behaviour of derived class.

In Fig.1.5, the Santro is a part of the class Hyundai which is again part of the class car and car is the part of the class vehicle. That means vehicle class is the parent class.

## **6. Polymorphism**

(Poly means —many and morph means —form). Polymorphism means the ability to take more than one form. Polymorphism plays a main role in allocate objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific activities associated with each operation may differ. Polymorphism is broadly used in implementing inheritance.

It means objects that can take on or assume many different forms. Polymorphism means that the same operations may behave differently on different classes. Booch defines polymorphism as the relationship of objects many different classes by some common super class. Polymorphism allows us to write generic, reusable code more easily, because we can specify general instructions and delegate the implementation detail to the objects involved.

For Example:

In a pay roll system, manager, office staff and production worker objects all will respond to the compute payroll message, but the real operations performed are object particular.

## **7. Dynamic Binding**

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code related with a given procedure call is not known until the time of the call at run time.

Dynamic binding is associated polymorphism and inheritance.

### **1.4 JAVA Features:**

As we know that the Java is an object oriented programming language developed by Sun Microsystems of USA in 1991. Java is first programming language which is not attached with any particular hardware or operating system. Program developed in Java can be executed anywhere and on any system.

Features of Java are as follows:

1. Compiled and Interpreted
2. Platform Independent and portable
3. Object- oriented
4. Robust and secure
5. Distributed
6. Familiar, simple and small
7. Multithreaded and Interactive
8. High performance
  
9. Dynamic and Extensible

## **1. Compiled and Interpreted**

Basically a computer language is either compiled or interpreted. Java comes together both these approach thus making Java a two-stage system.

Java compiler translates Java code to Bytecode instructions and Java Interpreter generate machine code that can be directly executed by machine that is running the Java program.

## **2. Platform Independent and portable**

Java supports the feature portability. Java programs can be easily moved from one computer system to another and anywhere. Changes and upgrades in operating systems, processors and system resources will not force any alteration in Java programs. This is reason why Java has become a trendy language for programming on Internet which interconnects different kind of systems worldwide. Java certifies portability in two ways.

First way is, Java compiler generates the bytecode and that can be executed on any machine. Second way is, size of primitive data types are machine independent.

## **3. Object-oriented**

Java is truly object-oriented language. In Java, almost everything is an Object. All program code and data exist in objects and classes. Java comes with an extensive set of classes; organize in packages that can be used in program by Inheritance. The object model in Java is trouble-free and easy to enlarge.

## **4. Robust and secure**

Java is a most strong language which provides many securities to make certain reliable code. It is design as garbage –collected language, which helps the programmers virtually from all memory

management problems. Java also includes the concept of exception handling, which detain serious errors and reduces all kind of threat of crashing the system.

Security is an important feature of Java and this is the strong reason that programmer use this language for programming on Internet.

The absence of pointers in Java ensures that programs cannot get right of entry to memory location without proper approval.

### **5. Distributed**

Java is called as Distributed language for construct applications on networks which can contribute both data and programs. Java applications can open and access remote objects on Internet easily. That means multiple programmers at multiple remote locations to work together on single task.

### **6. Simple and small**

Java is very small and simple language. Java does not use pointer and header files, goto statements, etc. It eliminates operator overloading and multiple inheritance.

### **7. Multithreaded and Interactive**

Multithreaded means managing multiple tasks simultaneously. Java maintains multithreaded programs. That means we need not wait for the application to complete one task before starting next task. This feature is helpful for graphic applications.

## 8. High performance

Java performance is very extraordinary for an interpreted language, majorly due to the use of intermediate bytecode. Java architecture is also designed to reduce overheads during runtime. The incorporation of multithreading improves the execution speed of program.

## 9. Dynamic and Extensible

Java is also dynamic language. Java is capable of dynamically linking in new class, libraries, methods and objects. Java can also establish the type of class through the query building it possible to either dynamically link or abort the program, depending on the reply.

Java program is support functions written in other language such as C and C++, known as native methods.

### 1.5 Comparison in Java and C++

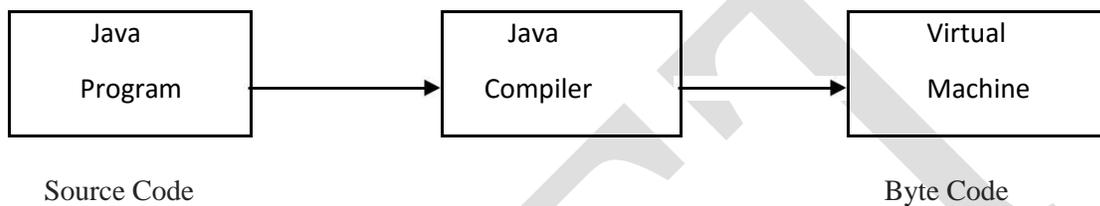
	Java	C++
1	Java is true Object-oriented language.	C++ is basically C with Object-oriented extension.
2	Java does not support operator overloading.	C++ supports operator overloading.
3	It supports labels with loops and statement blocks	It supports goto statement.
4	Java does not have template classes as in C++.	C++ has template classes.
5	Java compiled into byte code for the <a href="#">Java Virtual Machine</a> . The source code is independent on	Source code can be written to be platform independent and written to take advantage of platform.C++ typically compiled into

	<a href="#">operating system.</a>	machine code.
6	Java does not support multiple inheritance of classes but it supports interface.	C++ supports multiple inheritance of classes.
7	Runs in a protected virtual machine.	Exposes low-level system facilities.
8	Java does not support global variable. Every variable should declare in class.	C++ support global variable.
9	Java does not use pointer.	C++ uses pointer.
10	It Strictly enforces an object oriented <a href="#">programming paradigm.</a>	It Allows both <a href="#">procedural programming</a> and <a href="#">object-oriented programming.</a>
11	There are no header files in Java.	We have to use header file in C++.

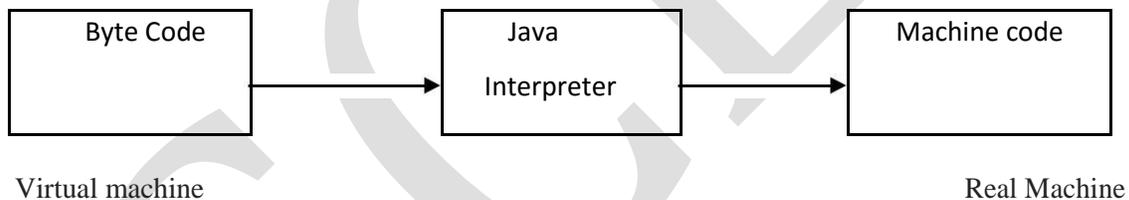
### 1.6 Java Virtual machine:

As we know that all programming language compilers convert the source code to machine code. Same job done by Java Compiler to run a Java program, but the difference is that Java compiler convert the source code into Intermediate code is called as bytecode. This machine is called the *Java Virtual machine* and it exists only inside the computer memory.

Following figure shows the process of compilation.



The Virtual machine code is not machine specific. The machine specific code is generated. By Java interpreter by acting as an intermediary between the virtual machine and real machines shown below



### **1.7 Java Environment:**

Java environment includes a number of development tools, classes and methods. The development tools are part of the system known as Java Development Kit (JDK) and the classes and methods are part of the Java Standard Library (JSL), also known as the Application Programming Interface (API).

Java Development kit (JDK) – The JDK comes with a set of tools that are used for developing and running Java program. It includes:

1. Appletviewer( It is used for viewing the applet)
2. Javac(It is a Java Compiler)
3. Java(It is a java interpreter)
4. Javap(Java diassembler,which convert byte code into program description)
5. Javah(It is for java C header files)
6. Javadoc(It is for creating HTML document)
7. Jdb(It is Java debugger)

For compiling and running the program we have to use following commands:

a) javac (Java compiler)

In java, we can use any text editor for writing program and then save that program with

—.java extension. Java compiler convert the source code or program in bytecode and interpreter convert —.java file in —.class file.

Syntax:

C:\javac filename.java

If my filename is —abc.java then the syntax will be

C:\javac abc.java

b) java(Java Interpreter)

As we learn that, we can use any text editor for writing program and then save that program with —.java extension. Java compiler convert the source code or program in bytecode and interpreter convert —.java file in —.class file.

Syntax:

C:\java filename

If my filename is abc.java then the syntax will be

C:\java abc

### **1.5 Simple Java Program:**

```
class FirstProgram
{
    public static void main(String args[])
    {
        System.out.println("—This is my first program");
    }
}
```

- The file must be named —FirstProgram.java to equivalent the class name containing the main method.
- 
- Java is case sensitive. This program defines a class called —FirstProgram.

A class is an object oriented term. It is designed to perform a specific task. A Java class is defined by its class name, an open curly brace, a list of methods and fields, and a close curly brace.

- 
- The name of the class is made of alphabetical characters and digits without spaces, the first character must be alphabetical.
- The line —public static void main (String [] args ) shows where the program will start running. The word main means that this is the main method –

- The JVM starts running any program by executing this method first. The
- main method in `—FirstProgram.java` consists of a single statement
- ```
System.out.println("This is my first program");
```

The statement outputs the character between quotes to the console.

Above explanation is about how to write program and now we have to learn where to write program and how to compile and run the program.

For this reason, the next explanation is showing the steps.

1. Edit the program by the use of Notepad.
2. Save the program to the hard disk.
3. Compile the program with the `javac` command.(Java compiler)
4. If there are syntax errors, go back to Notepad and edit the program.
5. Run the program with the `java` command.(Java Interpreter)
6. If it does not run correctly, go back to Notepad and edit the program.
7. When it shows result then stop.

## 2.1 Data types:

- A **data type** is a scheme for representing values. An example is `int` which is the Integer, a data type.

Values are not just numbers, but any manner of data that a computer can process. The data type defines the kind of data that is represented by a variable.

As with the keyword `class`, Java data types are case sensitive.

There are two types of data types

-

primitive data type non-

- primitive data type

In primitive data types, there are two categories

- 
- numeric means Integer, Floating points

Non-numeric means Character and Boolean

In non-primitive types, there are three categories

- classes
  - 
  - arrays
- interface

Following table shows the datatypes with their size and ranges.

| Data type | Size (byte) | Range                           |
|-----------|-------------|---------------------------------|
| byte      | 1           | -128 to 127                     |
| boolean   | 1           | True or false                   |
| char      | 2           | A-Z,a-z,0-9,etc.                |
| short     | 2           | -32768 to 32767                 |
| Int       | 4           | (about) -2 million to 2 million |
| long      | 8           | (about) -10E18 to 10E18         |
| float     | 4           | -3.4E38 to 3.4E18               |
| double    | 8           | -1.7E308 to 1.7E308             |

Fig: Datatypes with size and range

- 

### **2.1.1 Integer data type:**

Integer datatype can hold the numbers (the number can be positive number or negative number). In Java, there are four types of integer as follows:

- byte
- short int
- long
- 

We can make integer long by adding `_l` or `_L` at the end of the number.

### **2.1.2 Floating point data type:**

It is also called as Real number and when we require accuracy then we can use it.

There are two types of floating point data type.

float

double

It is represent single and double precision numbers. The float type is used for single precision and it uses 4 bytes for storage space. It is very useful when we require accuracy with small degree of precision. But in double type, it is used for double precision and uses 8 bytes of storage space. It is useful for large degree of precision.

### **2.1.3 Character data type:**

It is used to store single character in memory. It uses 2 bytes storage space.

### **2.1.4 Boolean data type:**

It is used when we want to test a particular condition during the execution of the program. There are only two values that a boolean type can hold: true and false.

Boolean type is denoted by the keyword `boolean` and uses only one bit of storage.

## 2.2 Mixing Data types:

Java allows mixing of constants and variables of different types in an expression, but during assessment it holds to very strict rules of type conversion.

When a computer considers an operand and an operator and if the operands are different types, the type is automatically converted to a higher type.

The following table shows the automatic type conversion.

|               | <b>char</b> | <b>byte</b> | <b>short</b> | <b>int</b> | <b>long</b> | <b>float</b> | <b>double</b> |
|---------------|-------------|-------------|--------------|------------|-------------|--------------|---------------|
| <b>Char</b>   | int         | int         | int          | int        | long        | float        | double        |
| <b>Byte</b>   | int         | int         | int          | int        | long        | float        | double        |
| <b>Short</b>  | int         | int         | int          | int        | long        | float        | double        |
| <b>Int</b>    | int         | int         | int          | int        | long        | float        | double        |
| <b>Long</b>   | long        | long        | long         | long       | long        | float        | double        |
| <b>Float</b>  | float       | float       | float        | float      | float       | float        | double        |
| <b>double</b> | double      | double      | double       | double     | double      | double       | double        |

## 2.3 Variables:

Variables are labels that express a particular position in memory and connect it with a data type.

The first way to declare a variable: This specifies its data type, and reserves memory for it. It assigns zero to primitive types and null to objects.

**dataType variableName;**

The second way to declare a variable: This specifies its data type, reserves memory for it, and puts an initial value into that memory. The initial

value must be of the correct data type.

**dataType variableName = initialValue;**

The first way to declare two variables: all of the same data type, reserves memory for each.

**dataType variableNameOne, variableNameTwo;**

The second way to declare two variables: both of the same data type, reserves memory, and puts an initial value in each variable.

**dataType variableNameI = initialValueI, variableNameII=initialValueII;**

### 2.3.1 Variable name:

- Use only the characters `_a'` through `_z'`, `_A'` through `_Z'`, `_0'` through `_9'`, character `__'`, and character `_$'`.
- 
- A name cannot include the space character. Do
- not begin with a digit.

A name can be of any realistic length.

- 
- 
-

Upper and lower case count as different characters. A name cannot be a reserved word (keyword).

A name must not previously be in utilized in this block of the program.

## **2.4 Constant:**

Constant means fixed value which is not change at the time of execution of program. In Java, there are two types of constant as follows:

- Numeric Constants
  - Integer constant
  - Real constant
- Character Constants
  - Character constant
  - String constant

### **2.4.1 Integer Constant:**

An Integer constant refers to a series of digits. There are three types of integer as follows:

- a) Decimal integer

Embedded spaces, commas and characters are not alloed in between digits. For example:

23 411

7,00,000

17.33

b) Octal integer

It allows us any sequence of numbers or digits from 0 to 7 with leading 0 and it is called as Octal integer.

For example:

011

00

0425

c) Hexadecimal integer

It allows the sequence which is preceded by 0X or 0x and it also allows alphabets from `_A'` to `_F'` or `_a'` to `_f'` (`_A'` to `_F'` stands for the numbers `_10'` to `_15'`) it is called as Hexadecimal integer.

For example:

0x7

00X

0A2B

### 2.4.2 Real Constant

It allows us fractional data and it is also called as floating point constant. It is used for percentage, height and so on.

For example:

0.0234 0.777 -

1.23

### 2.4.3 Character Constant

It allows us single character within pair of single quote.

For example:

'A'

'7'

'\'

### 2.4.4 String Constant

It allows us the series of characters within pair of double quote.

For example:

"WELCOME"

"END OF PROGRAM"

"BYE ...BYE"

"A"

### 2.4.5 Symbolic constant:

In Java program, there are many things which is requires repeatedly and if we want to make changes then we have to make these changes in whole program where this variable is used. For this purpose, Java provides 'final' keyword to declare the value of variable as follows:

Syntax:

final type Symbolic\_name=value;

For example:

If I want to declare the value of `__PI` then:

```
final float PI=3.1459
```

the condition is, Symbolic\_name will be in capital letter( it shows the difference between normal variable and symbolic name) and do not declare in method.

#### 2.4.6 Backslash character constant:

Java support some special character constant which are given in following table.

| Constant         | Importance   |
|------------------|--------------|
| <code>__b</code> | Back space   |
| <code>__t</code> | Tab          |
| <code>__n</code> | New line     |
| <code>__</code>  | Backslash    |
| <code>__</code>  | Single quote |
| <code>__</code>  | Double quote |

#### 2.5 Comments:

A **comment** is a note written to a human reader of a program. The program compiles and runs exactly the same with or without comments. Comments start

with the two characters `—//` (slash slash). Those characters and everything that follows them on the same line are ignored by the java compiler.

everything between the two characters `—/*` and the two characters `—*/` are unobserved by the compiler. There can be many lines of comments between the `—/*` and the `—*/`.

#### 2.6 Command line arguments:

Command line arguments are parameters that are supplied to the application program at the time of invoking its execution. They must be supplied at the time of its execution following the file name.

In the main () method, the args is confirmed as an array of string known as string objects. Any argument provided in the command line at the time of program execution, are accepted to the array args as its elements. Using index or subscripted entry can access the individual elements of an array. The number of element in the array args can be getting with the length parameter.

### **3.1 Introduction:**

A Java program is basically a set of classes. A class is defined by a set of declaration statements and methods or functions. Most statements contain expressions, which express the actions carried out on information or data. Smallest individual thing in a program are known as tokens. The compiler recognizes them for building up expression and statements.

### **3.2 Tokens in Java:**

There are five types of token as follows:

1. Literals
2. Identifiers
3. Operators
4. Separators

#### **3.2.1 Literals:**

Literals in Java are a sequence of characters (digits, letters and other characters) that characterize constant values to be stored in variables. Java language specifies five major types of literals are as follows:

1. Integer literals
2. Floating point literals
3. Character literals
4. String literals

## 5. Boolean literals

### 3.2.2 Identifiers:

Identifiers are programmer-created tokens. They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program. Java identifiers follow the following rules:

1. They can have alphabets, digits, and the underscore and dollar sign characters.
2. They must not start with a digit.
3. Uppercase and lowercase letters are individual.
4. They can be of any length.

Identifier must be meaningful, easily understandable and descriptive.

For example:

Private and local variables like `—lengthl`.

Name of public methods and instance variables begin with lowercase letter like `—additionall`

### 3.2.3 Keywords:

Keywords are important part of Java. Java language has reserved 50 words as keywords.

Keywords have specific meaning in Java. We cannot use them as variable, classes and method.

Following table shows keywords.

|          |          |        |            |
|----------|----------|--------|------------|
| abstract | char     | catch  | boolean    |
| default  | finally  | do     | implements |
| if       | long     | throw  | private    |
| package  | static   | break  | double     |
| this     | volatile | import | protected  |
| class    | throws   | byte   | else       |
| float    | final    | public | transient  |

|           |            |       |              |
|-----------|------------|-------|--------------|
| native    | instanceof | case  | extends      |
| int       | null       | const | new          |
| return    | try        | for   | switch       |
| interface | void       | while | synchronized |
| short     | continue   | goto  | super        |
| assert    | const      |       |              |

### 3.2.4 Operator:

Java carries a broad range of operators. An operator is symbols that specify operation to be performed may be certain mathematical and logical operation. Operators are used in programs to operate data and variables. They frequently form a part of mathematical or logical expressions. Categories of operators are as follows:

1. Arithmetic operators
2. Logical operators
3. Relational operators
4. Assignment operators
5. Conditional operators
6. Increment and decrement operators
7. Bit wise operators

#### 3.2.4.1 Arithmetic operators:

Arithmetic operators are used to make mathematical expressions and the working out as same in algebra. Java provides the fundamental arithmetic operators. These can operate on built in data type of Java.

Following table shows the details of operators.

| Operator | Importance/ significance |
|----------|--------------------------|
|----------|--------------------------|

|   |                              |
|---|------------------------------|
| + | Addition                     |
| - | Subtraction                  |
| / | Division                     |
| * | Multiplication               |
| % | Modulo division or remainder |

Now the following programs show the use of arithmetic operators.

### 3.2.4.2 Logical operators:

When we want to form compound conditions by combining two or more relations, then we can use logical operators.

Following table shows the details of operators.

| Operators | Importance/ significance |
|-----------|--------------------------|
|           | Logical – OR             |
| &&        | Logical –AND             |
| !         | Logical –NOT             |

The logical expression defer a value of true or false. Following table shows the truth table of Logical – OR and Logical – AND.

Truth table for Logical – OR operator:

| Operand1 | Operand3 | Operand1    Operand3 |
|----------|----------|----------------------|
| T        | T        | T                    |
| T        | F        | T                    |
| F        | T        | T                    |
| F        | F        | F                    |

T - True

F - False

Truth table for Logical – AND operator:

| Operand1 | Operand3 | Operand1 && Operand3 |
|----------|----------|----------------------|
| T        | T        | T                    |
| T        | F        | F                    |
| F        | T        | F                    |
| F        | F        | F                    |

T - True

F – False

Now the following program shows the use of Logical operators.

```
class LogicalOptr
{

public static void main (String args[])
{

boolean a = true;
boolean b = false;

System.out.println("a||b = " +(a||b));
System.out.println("a&&b = " +(a&&b));

System.out.println("a! = " +(!a));
}
}
```

Output:  $a||b =$

true

$a\&\&b =$  false  $a! =$

false

### 3.2.4.3 Relational Operators:

When evaluation of two numbers is performed depending upon their relation, assured decisions are made.

The value of relational expression is either true or false.

If  $A=7$  and  $A < 10$  is true while  $10 < A$  is false.

Following table shows the details of operators.

| Operator | Importance/ significance |
|----------|--------------------------|
| >        | Greater than             |
| <        | Less than                |
| !=       | Not equal to             |
| >=       | Greater than or equal to |
| <=       | Less than or equal to    |

Now, following examples show the actual use of operators.

- 1) If  $10 > 30$  then result is false
- 2) If  $40 > 17$  then result is true
- 3) If  $10 >= 300$  then result is false
- 4) If  $10 <= 10$  then result is true

Now the following program shows the use of operators.

(1) Program 1:

```
class Reloptr1  
  
{  
  
    public static void main (String args[])  
  
    {  
  
        int a = 10; int  
        b = 30;  
  
        System.out.println("a>b = " +(a>b));  
        System.out.println("a<b = " +(a<b));  
        System.out.println("a<=b = " +(a<=b));  
  
    }  
}
```

Output: a>b =  
false a<b = true  
a<=b = true

(2) Program 3

```
class Reloptr3  
  
{  
  
    public static void main (String args[])  
  
    {
```

```

int a = 10; int
b = 30; int c
= 30;

System.out.println("a>b = " +(a>b));
System.out.println("a<b = " +(a<b));
System.out.println("a<=c = " +(a<=c));
System.out.println("c>b = " +(c>b));
System.out.println("a<c = " +(a<c));
System.out.println("b<=c = " +(b<=c));

}
}

```

Output: a>b =  
false a<b = true

a<=c = true c>b  
= true a<c =  
true b<=c = true

#### 3.2.4.4 Assignment Operators:

Assignment Operators is used to assign the value of an expression to a variable and is also called as Shorthand operators.

Variable\_name binary\_operator = expression

Following table show the use of assignment operators.

| Simple Assignment Operator | Statement with shorthand Operators |
|----------------------------|------------------------------------|
|----------------------------|------------------------------------|

|           |          |
|-----------|----------|
| A=A+1     | A+=1     |
| A=A-1     | A-=1     |
| A=A/(B+1) | A/=(B+1) |
| A=A*(B+1) | A*=(B+1) |
| A=A/C     | A/=C     |
| A=A%C     | A%=C     |

These operators avoid repetition, easier to read and write.

Now the following program shows the use of operators.

```
class Assoptr
{
    public static void main (String args[])
    {
        int a = 10; int
        b = 30; int c
        = 30;

        a+=1; b-
        =3;
        c*=7;

        System.out.println("a = "+a);
        System.out.println("b = "+b);
        System.out.println("c = "+c);

    }
}
```

Output: a  
= 11 b =  
18 c = 310

### 3.2.4.5 Conditional Operators:

The character pair ?: is a ternary operator of Java, which is used to construct conditional expressions of the following form:

Expression1 ? Expression3 : Expression3

The operator ?: works as follows:

Expression1 is evaluated if it is true then Expression3 is evaluated and becomes the value of the conditional expression. If Expression1 is false then Expression3 is evaluated and its value becomes the conditional expression.

### 3.2.4.6 Increment and Decrement Operators:

The increment operator ++ adds 1 to a variable. Usually the variable is an integer type, but it can be a floating point type. The two plus signs must not be split by any character. Usually they are written immediately next to the variable.

Following table shows the use of operators.

| Expression | Process                            | Example               | end result   |
|------------|------------------------------------|-----------------------|--------------|
| A++        | Add 1 to a variable<br>after use.  | int A=10,B;<br>B=A++; | A=11<br>B=10 |
| ++A        | Add 1 to a variable<br>before use. | int A=10,B;<br>B=++A; | A=11<br>B=11 |
| A--        | Subtract 1 from a                  | int A=10,B;           | A=9          |

|     |                                        |                       |            |
|-----|----------------------------------------|-----------------------|------------|
|     | variable after use.                    | B=A--;                | B=10       |
| --A | Subtract 1 from a variable before use. | int A=10,B;<br>B=--A; | A=9<br>B=9 |

Now the following program shows the use of operators.

### 3.2.4.7 Bit Wise Operators:

Bit wise operator execute single bit of their operands. Following table shows bit wise operator:

| Operator | Importance/ significance |
|----------|--------------------------|
|          | Bitwise OR               |
| &        | Bitwise AND              |
| &=       | Bitwise AND assignment   |
| =        | Bitwise OR assignment    |
| ^        | Bitwise Exclusive OR     |
| <<       | Left shift               |
| >>       | Right shift              |
| ~        | One's complement         |

Separators are symbols. It shows the separated code. they describe function of our code.

| Name | use                                                                                                                               |
|------|-----------------------------------------------------------------------------------------------------------------------------------|
| ()   | Parameter in method definition, containing statements for conditions, etc.                                                        |
| { }  | It is used for define a code for method and classes                                                                               |
| []   | It is used for declaration of array                                                                                               |
| ;    | It is used to show the separate statement                                                                                         |
| ,    | It is used to show the separation in identifier in variable declaration                                                           |
| .    | It is used to show the separate package name from sub-packages and classes, separate variable and method from reference variable. |

### 3.3 Operator Precedence in Java:

An arithmetic expression without any parentheses will be calculated from left to right using the rules of precedence of operators.

There are two priority levels of arithmetic operators are as follows:

- (a) High priority (\* / %)
- (b) Low priority (+ -)

The evaluation process includes two left to right passes through the expression. During the first pass, the high priority operators are applied as they are encountered.

During the second pass, the low priority operators are applied as they are encountered.

For example:

$$Z=A-B/3+C*3-1$$

When A=10, B=13, C=3

First pass:

$$Z=10-(13/3) + (3*3)-1$$

$$Z=10-4+3-1$$

Second pass:

$$Z=6+3-1$$

$$Z=7$$

Answer is=7

### Control Structure:

In java program, control structure is can divide in three parts:

- Selection statement
-

Iteration statement

- 

Jumps in statement

#### **4.2.1 Selection Statement:**

Selection statement is also called as Decision making statements because it provides the decision making capabilities to the statements.

In selection statement, there are two types:

- if statement
- switch statement

These two statements are allows you to control the flow of a program with their conditions.

##### **4.2.1.1 if Statement:**

The “if statement” is also called as conditional branch statement. It is used to program execution through two paths. The syntax of “if statement” is as follows:

Syntax:

```
if (condition)
```

```
{
```

```
Statement 1;
```

Statement 2;

...

}

else

{

Statement 3;

Statement 4;

...

}

The “if statement” is a commanding decision making statement and is used to manage the flow of execution of statements. The “if statement” is the simplest one in decision statements. Above syntax is shows two ways decision statement and is used in combination with statements.

Following figure shows the “if statement”

#### **4.2.1.1.1 Simple if statement:**

In statement block, there may be single statement or multiple statements. If the condition is true then statement block will be executed. If the condition is false then statement block will omit and statement-a will be executed.

### **.2.1.1.2 The if...else statement:**

Syntax:

If (condition)

{

True - Statement block;

}

else

{

False - Statement block;

}

Statement-a;

If the condition is true then True - statement block will be executed. If the condition is false then False - statement block will be executed. In both cases the statement-a will always executed.

Following program shows the use of if statement.

### **Nesting of if-else statement:**

Syntax:

```
if (condition1)
{
    If(condition2)
        {
            Statement block1;
        }
    else
        {
            Statement block2;
        }
}
else
{
```

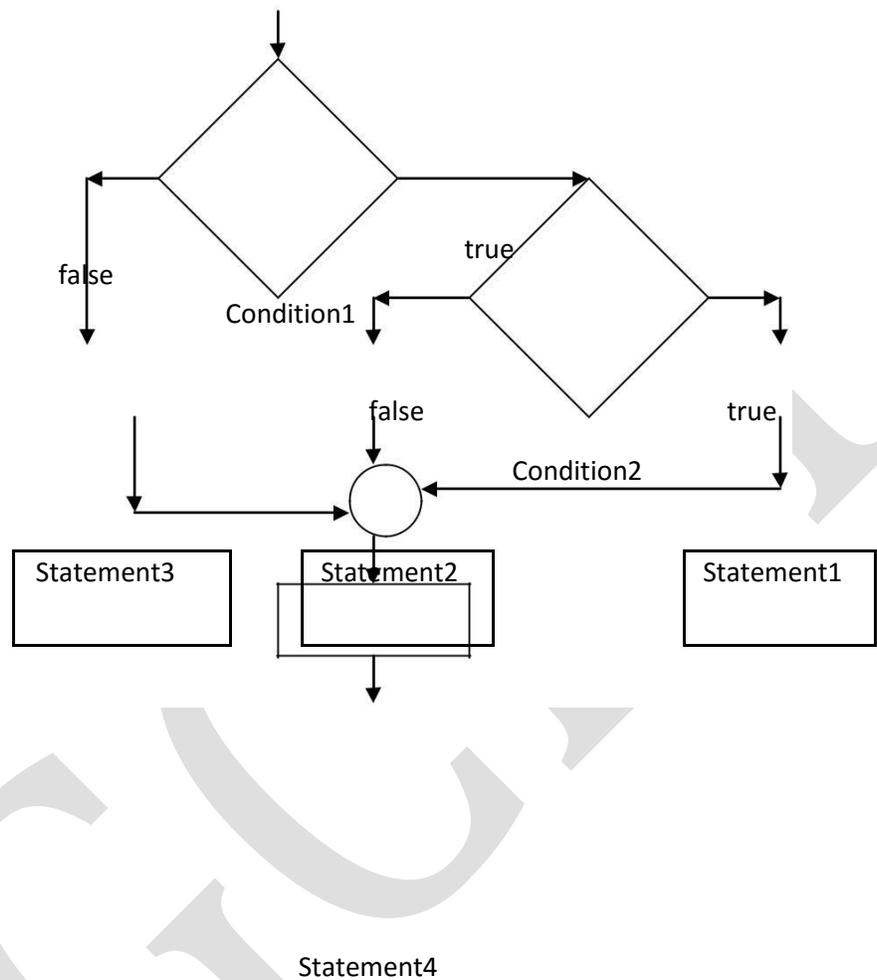
Statement block3;

}

Statement 4;

GCCEFT

If the condition1 is true then it will be goes for condition2. If the condition2 is true then statement block1 will be executed otherwise statement2 will be executed. If the condition1 is false then statement block3 will be executed. In both cases the statement4 will always executed.



For example:Write a program to find out greatest number from three numbers.

```
class greatest
```

```
{  
  
public static void main(String args[])
```

```
{
```

```
int a=10;
```

```
int b=20;
```

```
int c=3;
```

```
if(a>b)
```

```
{
```

```
if(a>c)
```

```
{  
    System.out.println("a is greater number");  
}
```

else

```
{  
    System.out.println("c is greater number");  
}
```

```
}
```

else

```
{
```

```
    if(c>b)
```

```
    {
```

```
        System.out.println("c is greater number");
```

```
    }
```

else

```
{
```

```
    System.out.println("b is greater number");
```

```
        }  
    }  
}
```

```
Output: C:\MCA>java  
greatest b is greater  
number
```

GCEFT

#### 4.2.1.2 switch statement:

In Java, switch statement check the value of given variable or statement against a list of case values and when the match is found a statement-block of that case is executed. Switch statement is also called as multiway decision statement.

Syntax:

```
switch(condition)// condition means case value  
  
{  
  
    case value-1:statement block1;break;  
  
    case value-2:statement block2;break;  
  
    case value-3:statement block3;break;  
  
    ...  
  
    default:statement block-default;break;  
  
}  
  
statement a;
```

The condition is byte, short, character or an integer. value-1,value-2,value-3,...are constant and is called as labels. Each of these values be matchless or unique with the statement. Statement block1, Statement block2, Statement block3,..are list of statements which contain one statement or more than one statements. Case label is always end with ":" (colon).

#### 4.2.2 Iteration Statement:

The process of repeatedly executing a statements and is called as looping. The statements may be executed multiple times (from zero to infinite number). If a loop executing continuous then it is called as Infinite loop. Looping is also called as iterations.

In Iteration statement, there are three types of operation:

- for
- do-while
- while

#### 4.2.2.1 for loop:

The for loop is entry controlled loop. It means that it provide a more concious loop control structure.

Syntax:

```
for(initialization;condition;iteration)//iteration means increment/decrement
{
Statement block;
}
```

When the loop starts, first part (i.e. initialization) is executed. It is just like a counter and provides the initial value of loop. But the thing is, initialization is executed only once. The next part (i.e. condition) is executed after the initialization. The important thing is, this part provides the condition for looping. If the condition is satisfying then loop will execute otherwise it will terminate.

Third part (i.e. iteration) is executed after the condition. The statements that increment or decrement the loop control variables.

Here we declare  $i=1$  and then it checks the condition that if  $i < 10$  then only loop will be executed. After first iteration the value of  $i$  will print and it will be incremented by 1. Now the value of  $i=2$  and again we have to check the condition and value of  $i$  will print and then increment  $i$  by 1 and so on.

#### **4.2.2.2 while loop:**

The while loop is an entry-controlled loop statement. The condition is evaluated, if the condition is true then the block of statements or statement block is executed otherwise the block of statement is not executed.

Syntax:

```
While(condition)
```

```
{
```

```
Statement block;
```

```
}
```

#### 4.2.2.3 do-while loop:

In do-while loop, first attempt of loop should be execute then it check the condition.

The benefit of do-while loop/statement is that we get entry in loop and then condition will check for very first time. In while loop, condition will check first and if condition will not satisfied then the loop will not execute.

Syntax:

```
do
```

```
{
```

```
Statement block;
```

```
}
```

```
While(condition);
```

In program,when we use the do-while loop, then in very first attempt, it allows us to get enter in loop and execute that loop and then check the condition.

Following program show the use of do-while loop.

## CLASSES

- How to create class
- 
-

How to create method

How to create constructor

## 5.2 class

***Definition:** A class is a collection of objects of similar type. Once a class is defined, any number of objects can be produced which belong to that class.*

### Class Declaration

```
class classname
```

```
{
```

```
...
```

```
ClassBody
```

```
...
```

```
}
```

Objects are instances of the Class. Classes and Objects are very much related to each other. Without objects you can't use a class.

A general class declaration:

```
class name1
```

```
{
```

```
//public variable
```

```
declaration void
```

```
methodname()
```

```
{  
//body of method...  
//Anything  
}  
}
```

Now following example shows the use of method.

```
class Demo  
{  
  
private  
int x,y,z;  
public  
void  
input()  
{  
  
x=10;  
y=15;  
}  
  
public void sum()  
{  
z=x+y;  
}  
public void print_data()  
{  
System.out.println("Answer is =|| +z);  
}
```

```
public static void main(String args[])
{

Demo object=new
Demo(); object.input();
object.sum();
object.print_data();

}
}
```

In program,

```
Demo object=new
Demo(); object.input();
object.sum();
object.print_data();
```

In the first line we created an object.

The three methods are called by using the dot operator. When we call a method the code inside its block is executed.

The dot operator is used to call methods or access them.

### **5.2.1 Creating “main” in a separate class**

We can create the main method in a separate class, but during compilation you need to make sure that you compile the class with the `—mainll` method.

```
class Demo
{
```

```
private int x,y,z;
public void input() {
x=10;

y=15;
}
public void sum()
{
z=x+y;
}
public void print_data()

{
System.out.println("Answer is =| +z);
}

}
class SumDemo
{
public static void main(String args[])
{

Demo object=new
Demo(); object.input();
object.sum();
object.print_data();
```

```
}  
}
```

### 5.2.3 use of dot operator

We can access the variables by using dot operator. Following program shows the use of dot operator.

```
class DotDemo  
{  
int x,y,z;  
  
public void  
sum(){ z=x+y;  
  
}  
  
public void show(){  
System.out.println("The Answer is "+z);  
  
}  
}  
class Demo1  
{  
  
public static void main(String args[]){  
DotDemo object=new DotDemo();  
DotDemo object2=new DotDemo();  
object.x=10;  
  
object.y=15;  
object2.x=5;
```

```
object2.y=10;
object.sum();
object.show();
object2.sum();
object2.show();
}}
```

### **output:**

```
C:\cc>javac Demo1.java
```

```
C:\cc>java Demo1
```

```
The Answer is 25
```

```
The Answer is 15
```

- **Instance Variable**

All variables are also known as instance variable. This is because of

the fact that each instance or object has its own copy of values for the variables.

Hence other use of the —*dot*” operator is to initialize the value of variable for that instance.

## 5.2.6 Method Overloading

Method overloading means method name will be same but each method should be different parameter list.

```
class prg1
{

int x=5,y=5,z=0;
public void sum()
{

z=x+y; System.out.println("Sum is
"+z);

}
public void sum(int a,int b)
{
x=a;

y=b;

z=x+y; System.out.println("Sum is
"+z);

}
public int sum(int a)
{
x=a;
```

```
z=x+y;
return z;
}
}
class Demo
{
public static void main(String args[])
{

prg1 obj=new prg1(); obj.sum(); obj.sum(10,12);
System.out.println(+obj.sum(15));
}
}
```

Output: sum is 10 sum is 22 27

### 5.2.7 Passing Objects as Parameters

Objects can even be passed as parameters. class para123

```
{
int n,n2,sum,mul;
public void take(int x,int y)
{
n=x;
n2=y;
}
public void sum()
{
sum=n+n2;
System.out.println("The Sum is"+sum);
}
```

```
public void take2(para123 obj)
{
n=obj.n;
n2=obj.n2;
}
public void multi()
{

mul=n*n2; System.out.println("Product is"+mul);

}
}
class DemoPara
{
public static void main(String args[])
{

para123 ob=new para123(); ob.take(3,7);
ob.sum();
ob.take2(ob);
ob.multi();
}
}
```

Output:

```
C:\cc>javac DemoPara.java
```

```
C:\cc>java DemoPara
```

The Sum is10

Product is21

We have defined a method —*take2*” that declares an object named obj as parameter. We have passed ob to our method. The method —*take2*” automatically gets 3,7 as values for n and n2.

### 5.2.10 Passing Values to methods and Constructor:

These are two different ways of supplying values to methods. Classified under these two titles -

1.Pass by Value

2.Pass by Address or Reference

- **Pass by Value**-When we pass a data type like int, float or any other datatype to a method or some constant values like(15,10). They are all passed by value. A copy of variable’s value is passed to the receiving method and hence any changes made to the values do not affect the actual variables.

```
class Demopbv
{
int n,n2;
public void get(int x,int y)
{

x=x*x; //Changing the values of passed
arguments y=y*y; //Changing the values of
passed arguments

}
}
```

```
class Demo345
{
```

```
public static void main(String args[])
{

int
a,b;
a=1;
b=2;

System.out.println("Initial Values of a & b "+a+"
"+b); Demopbv obj=new Demopbv();
obj.get(a,b);
System.out.println("Final Values "+a+" "+b);
}
}
```

Output:

```
C:\cc>javac Demo345.java
```

```
C:\cc>java Demo345
```

```
Initial Values of a & b 1 2
```

```
Final Values 1 2
```

- **Pass by Reference**

Objects are always passed by reference. When we pass a value by reference, the reference or the memory address of the variables is passed. Thus any changes made to the argument causes a change in the values which we pass.

Demonstrating Pass by Reference---

```
class pass_by_ref
{
int n,n2;
```

```

public void get(int a,int b)
{
n=a;
n2=b;
}
public void doubleit(pass_by_ref temp)
{
temp.n=temp.n*2;
temp.n2=temp.n2*2;
}
}
class apply7
{
public static void main(String args[])
{
int x=5,y=10;

pass_by_ref obj=new pass_by_ref();
obj.get(x,y); //Pass by Value
System.out.println("Initial Values are-- ");
System.out.println(+obj.n);
System.out.println(+obj.n2); obj.doubleit(obj);
//Pass by Reference System.out.println("Final
Values are"); System.out.println(+obj.n);
System.out.println(+obj.n2);
}
}

```

## 5.2.9 ABSTRACT CLASSES

**Definition:** An abstract class is a class that is declared as abstract. It may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclass.

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

```
abstract void studtest(int rollno, double testfees);
```

If a class includes abstract methods, the class itself must be declared abstract, as in:

```
public abstract class GraphicObject
```

```
{
```

```
// declare fields
```

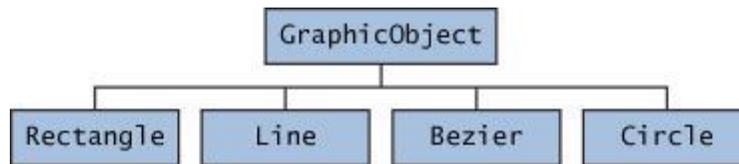
```
// declare non-abstract  
methods abstract void  
draw();
```

```
}
```

When an abstract class is subclass, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, the subclass must also be declared abstract.

**For example:** In an object-oriented drawing application, you can draw circles, rectangles, lines, Bezier curves, and many other graphic objects. These objects all have certain states (for example: position, orientation, line color, fill color) and behaviors (for example: moveTo, rotate, resize, draw) in common. Some of these states and behaviors are the same for all graphic objects—for example: position, fill color, and moveTo. Others require different implementations—for example, resize or draw. All

GraphicObjects must know how to draw or resize themselves; they just differ in how they do it. This is a perfect situation for an abstract superclass. You can take advantage of the similarities and declare all the graphic objects to inherit from the same abstract parent object—for example, GraphicObject, as shown in the following figure.



**How to implement above diagram concept with source code:**

```
abstract class GraphicObject
{
    int x, y;
    ...
    void moveTo(int newX, int newY)
{
    ...
}
```

```
    abstract void draw();

    abstract void resize();

}
```

Each non-abstract subclass of `GraphicObject`, such as `Circle` and `Rectangle`, must provide implementations for the `draw` and `resize` methods:

```
class Circle extends GraphicObject {
```

```
    void draw() {
```

```
        ...
```

```
    }
```

```
    void resize() {
```

```
        ...
```

```
    }
```

```
}
```

```
class Rectangle extends GraphicObject {
```

```
    void draw() {
```

```
        ...
```

```
    }  
  
    void resize() {  
  
        ...  
  
    }  
  
}
```

Abstract classes are those which can be used for creation of objects. However their methods and constructors can be used by the child or extended class. The need for abstract classes is that you can generalize the super class from which child classes can share its methods. The subclass of an abstract class which can create an object is called as "concrete class".

For example:

Abstract class

A

```
{  
  
    abstract void  
    method1(); void  
    method2()  
    {  
        System.out.println("this is real method");  
    }  
}
```

class B extends A

```
{  
    void method1()  
    {
```

```
System.out.println("B is execution of method1");
}
}
class demo
{
public static void main(String arg[])

{

B    b=new
B();
b.method1();
b.method2();

}
}
```

### **5.2.10 Extending the class:**

---

---

Inheritance allows to subclass or child class to access all methods and variables of parent class. Syntax:

```
class subclassname extends superclassname
{
Variables;
Methods;
.....
}
```

For example: calculate area and volume by using

Inheritance. class data

```
{
```

```
int l;  
int b;  
data(int c, int d)  
{  
l=c;  
b=d;  
}  
int area( )  
{  
return(l*b);  
}  
}  
class data2 extends data  
{  
int h;  
data2(int c,int d, int a)  
{  
super(c,d);  
h=a;  
}  
int volume()  
{  
return(l*b*h);  
}  
}  
class dataDemo  
{  
public static void main(String args[])  
{
```

```
data2 d1=new data2(10,20,30);
```

```
int area1=d1.area();//superclass method int
```

```
volume1=d1.volume( );// subclass method
```

```
System.out.println("Area="+area1);
```

GCEET

```
System.out.println("Volume="+volume1);  
}  
}
```

Output:

```
C:\cc>javac dataDemo.java
```

```
C:\cc>java dataDemo
```

```
Area=200
```

```
Volume=6000
```

"Is A" - is a subclass of a superclass (ex: extends)

"Has A" - has a reference to (ex: variable, ref to object).

- **Access Control** –

Away to limit the access others have to your code.

- **Same package** - can access each others' variables and methods, except for private members.
- **Outside package** - can access public classes. Next, can access members that are public. Also, can access protected members if the class is a subclass of that class.

Same package - use package keyword in first line of source file, or no package keyword and in same directory.

- **Keywords** -

1. public - outside of package access.
2. [no keyword] - same package access only.
3. protected - same package access. Access if class is a subclass of, even if in another package.

4. private - same class access only.

## Interfaces

### 6.1 Introduction

In chapter 5 you have learnt the following concepts:

- **Abstract** class, which allows you to create methods in a class without writing the code for execution of the method (implementation of the method).
- **Inheritance** through the keyword '**extends**' which tells the machine that an (inherited) class defined is of the type of a base class.
- Methods in the inherited class must provide implementation. (except when the inherited class is an **Abstract** class as well).

**Interface** takes the above concepts even further. It provides a mechanism to define a class with absolutely no implementation (code for execution of a method or logic).

### 6.2 More about 'interface'

One or more classes can **implement** a defined **interface**

When a class implements a defined interface, it has to implement (write the code, execution logic) for all the methods defined by the

#### Syntax of Interface

To define an interface, use the **interface** keyword instead of the **class** keyword.

SYNTAX:

```
package xxx.xxx;
interface MusicPlayer{

    // Cannot have method implementations: void on();

    void off(); void play(); void
    stop();
}
```

Points to note above:

- A semicolon after the method definition
- No implementation logic in the method above interface
- keyword instead of class

### 6.3 Access

In the above example, we've not defined whether the interface is public, private or protected. A private interface makes no sense. If not defined the above interface is visible in the package where the interface belongs. You can define an interface public – which means the interface is visible outside the package as well.

Methods inside the interface are public by default. So in the above example, the methods are public and visible outside of the package as well.

The class which inherits the methods must explicitly define the methods to be public.

SYNTAX:

```
class MP3Player implements MusicPlayer{

    public void on(){
```

```
        System.out.println("the MP3 Player is ON");

    }

    public void off(){

        System.out.println("the MP3 Player is OFF");
    }

    public void play(){

        System.out.println("the MP3 Player is playing");
    }

    public void stop(){

        System.out.println("the MP3 Player is off");
    }

}
```

### **Multiple Inheritance**

In Java, there is nothing which prevents from inheriting from multiple interfaces. Since there are no implementations in the methods (code in the methods), there is no danger or overwriting any implementations between multiple interfaces.

// Multiple interfaces.

```
interface MusicPlayer { void on();

    void off(); void play(); void stop();

}
```

```
interface VideoPlayer{ void on();void off(); void play(); void stop();

void changeContrast(int x); void changeBrightness(int x);
}
}
class iPod implements MusicPlayer, VideoPlayer{
public void on(){

    System.out.println("the MP3 Player is ON");
}
public void off(){
    System.out.println("the MP3 Player is OFF");
}
public void play(){
    System.out.println("the MP3 Player is playing");
}

public void stop(){

    System.out.println("the MP3 Player is off");

}

public void changeContrast(int x){ System.out.println("Constrast Changed by" + x);
}

public void changeBrightness(int x){

System.out.println("Brightnesss Changed by" + x);
```

SECRET

## Interfaces and Abstract Classes

Interfaces are similar to abstract classes. The differences are as follows:

1. All methods in an interface are abstract. Which means all methods must be empty; no code implemented.
2. In abstract class, the methods can have code/implementation within it. Atleast one method must be abstract.
3. All properties (data fields) in an interface are static final. Properties in an abstract class need not be static final.
4. Interfaces are implemented(implements keyword); Abstract classes are extended(extends keyword)
5. Class can extend only one abstract class; where as a class can implement multiple interfaces (multiple inheritance)
6. Contractual obligation: When a class specifies that it implements an interface, it must define all methods of that interface. A class can implement many different interfaces. If a class doesn't define all methods of the interfaces it agreed to define (by the implements clause), the compiler gives an error message, which typically says something like "This class must be declared abstract". An abstract class is one that doesn't implement all methods it said it would. The solution to this is almost always to implement the missing methods of the interface. A misspelled method name or incorrect parameter list is the usual cause, not that it should have been abstract!

### 6.6 inheritance within interfaces

In the above example, please note

- ElectronicDevices is an interface.
- MusicPlayer and VideoPlayer are interfaces which “**extend**” ElectronicDevices
- iPod is a class which implements MusicPlayer and VideoPlayer

So, if ElectronicDevices interface had one property – which is “powerSource”; it would be inherited by all classes which implement MusicPlayer or VideoPlayer

Example for practice:

Write a class that implements the CharSequence interface found in the java.lang package. Your implementation should return the string backwards. Select one of the sentences from this book to use as the data. Write a small main method to test your class; make sure to call all four methods.

Answer 1:

```
// CharSequenceDemo presents a String value -- backwards.

public class CharSequenceDemo implements CharSequence {

    private String s;
    public CharSequenceDemo(String s) {

        //It would be much more efficient to just reverse the string

        //in the constructor.

        this.s = s;

    }
}
```

```
private int fromEnd(int i) {
    return s.length() - 1 - i;
}

public char charAt(int i) {
    if ((i < 0) || (i >= s.length())) {
        throw new StringIndexOutOfBoundsException(i);
    }

    return s.charAt(fromEnd(i));
}

public int length() { return s.length();
}

public charSequence subSequence(int start, int end) { if
(start < 0) {

    throw new StringIndexOutOfBoundsException(start);

}

if (end > s.length()) {

    throw new StringIndexOutOfBoundsException(end);

}

if (start > end) {

    throw new StringIndexOutOfBoundsException(start - end);

}
```

```

StringBuilder sub =

    new StringBuilder(s.subSequence(fromEnd(end), fromEnd(start))); return sub.reverse();

}

public String toString() {

    StringBuilder s = new StringBuilder(this.s);
    return s.reverse().toString();
}

//Random int from 0 to max. private static
int random(int max) {
    return (int) Math.round(Math.random() * max + 0.5);
}

public static void main(String[] args) { CharSequenceDemo s =new CharSequenceDemo("Write a class that
    implements the CharSequence interface found in the java.lang package.");
    //exercise charAt() and length() for (int
    i = 0; i < s.length(); i++) {

        System.out.println(s.charAt(i));

    }
    //exercise subSequence() and length(); int start = random(s.length() - 1);

    int end = random(s.length() - 1 - start) + start;
    System.out.println(s.subSequence(start, end));
    //exercise toString();
    System.out.println(s);
}

```

}

## EXCEPTION HANDLING

**7.0 Objective:** In this lesson of Java Tutorial, you will learn...

GCEET

1. The exception handling mechanism.
2. Write try ... catch structures to catch expected exceptions
3. Use finally blocks to guarantee execution of code
4. Throw/ Throws exceptions

## 7.1 Introduction

An exception is an event, which occurs during the execution of the program, that an interrupt the normal flow of the program's instruction. In other words, Exceptions are generated when a recognized condition, usually an error condition, arises during the execution of a method. Java includes a system for running exceptions, by tracking the potential for each method to throw specific exceptions. For each method that could throw an exception, your code must report to the Java compiler that it could throw that exact exception. The compiler marks that method as potentially throwing that exception, and then need any code calling the method to handle the possible exception. Exception handling is basically use five keyword as follows:

- try
- catch
- throw
- throws
- finally

## 7.2 Overview

Exceptions are generated when an error condition occur during the execution of a method. It is possible that a statement might throw more than one kind of exception. Exception can be generated by Java-runtime system or they can be manually generated by code. Error-Handling becomes a necessary while

developing an application to account for exceptional situations that may occur during the program execution, such as

- Run out of memory
  
- Resource allocation Error
  
- Inability to find a file
  
- Problems in Network connectivity.

In this unit we will learn the exception handling mechanism.

### **7.3 What is Exceptions and handling exception?**

*Exceptions* are generated when a recognized an error condition during the execution of a program. Java includes a system for running exceptions, by tracking the potential for each method to throw specific exceptions

- for each method that could throw an exception, your code must report to the Java compiler that it could throw that exact exception.

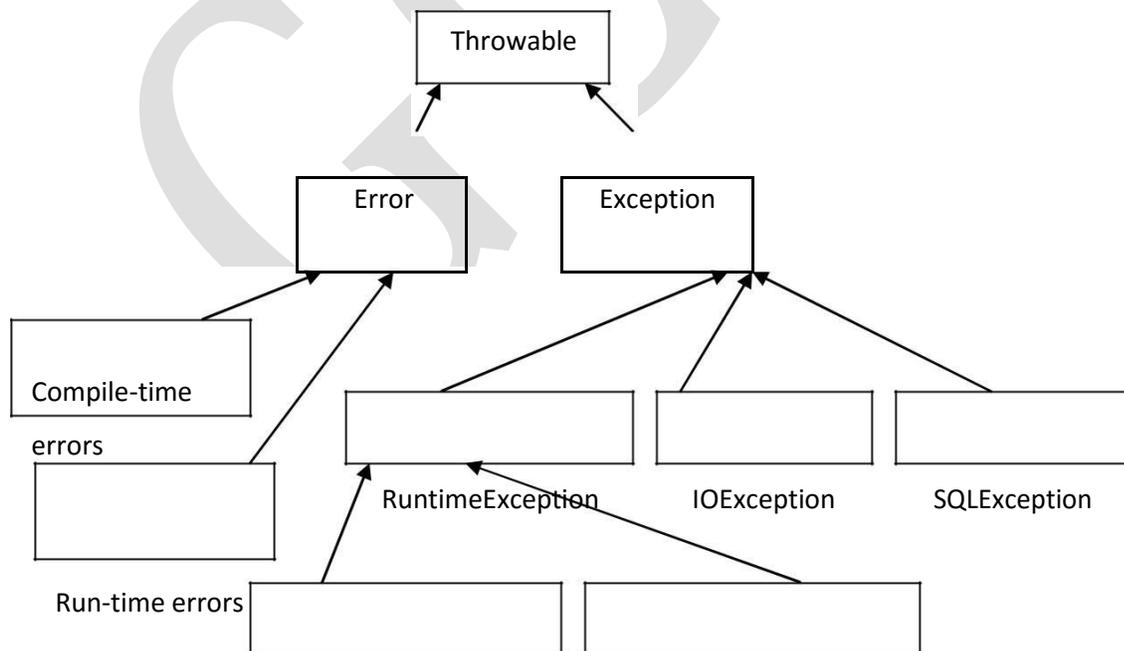
- the compiler marks that method as potentially throwing that exception, and then need any code calling the method to handle the possible exception.

There are two ways to handle an exception:

- you can try the "risky" code, catch the exception, and do something about it, after which the transmission of the exception come to an end
- you can mark that this method throws that exception, in which case the Java runtime engine will throw the exception back to the method.

So, if you use a method in your code that is marked as throwing a particular exception, the compiler will not allow that code unless you handle the exception. If the exception occurs in a try block, the JVM looks to the catch block(s) that follow to see if any of them equivalent the exception type. The first one that matches will be executed. If none match, then this methods ends, and execution jumps to the method that called this one, at the point the call was made.

Following figure shows the Exception type.



ArithmeticException

NullPointerException

Figure 7.1. A partial view of the Throwable family

An error means fault and there are two types of error as follows:

### **7.3.1 Compile time errors**

Compiler time error means Java compiler identify the syntax error at the time of compilation. And without successfully compilation, compiler does not create .class file. That means we have to compile the program which should be error free and then compiler creates .class file of the program and then we can run the program.

The common problems are:

- Missing braces
- Missing semicolon
- Missing double quote in string
- = instead of == operator
- And so on.

For example: class

Try1

```
{  
public static void main(String args[])  
{  
  
int a=12; int  
b=0; int c=a/b  
  
System.out.println("Division is+c);  
}  
}
```

Output:

```
C:\cc>javac Try1.java Try1.java:8: ';' expected  
System.out.println("Division  
is+c);
```

^

Try1.java:8: unclosed string literal

```
System.out.println("Division is+c);
```

^

2 errors

### 7.3.2 Run time errors

Several time program may compile successfully and compiler creates the .class file of the program but when the time of running the program, it shows the error and that type of error called run time error.

The common problems are:

- Divide by zero
- Conversion of invalid string to number
- access the element that is out of bound of an array
- Passing the parameters with invalid range.
- And so on.

For example: write a program to find out division of two numbers. class

Try1

```
{
```

```
public static void main(String args[])
{

int a=12;
int b=0;
int c=a/b;

System.out.println("Division is"+c);
}
}
```

Output:

```
C:\cc>javac Try1.java
```

```
C:\cc>java Try1
```

```
Exception in thread "main" java.lang.ArithmeticException: / by
zero at Try1.main(Try1.java:7)
```

### 7.3.3 try...catch:

If a method is going to resolve potential exception internally, the line of code that could generate the exception is placed inside a try block

- there may be other code inside the try block, before and/or after the risky line(s) - any code that depends upon the risky code's success should be in the try block, since it will automatically be skipped if the exception occurs

Syntax –

```
try
{
```

```
        code risky/unsafe
        code
        code that depends on the risky code succeeding
    }
```

There is usually at least one catch block immediately after the try block

- a catch block must specify what type of exception it will catch

Syntax –

```
catch (ExceptionClassName exceptionObjectName)
{
    code using methods from exceptionObjectName
}
```

-

- there can be more than one catch block, each one marked for a correct exception class

- the exception class that is caught can be any class in the exception hierarchy, either a general (base) class, or a very correct (derived) class

the catch block(s) must handle all checked exceptions that the try block is known to throw unless you want to throw that exception back to the method.

- it is possible to have a try block without any catch blocks if you have a finally block but any checked exceptions still need to be caught, or the method needs to declare that it throws them

If an exception occurs within a try block, execution jumps to the first catch block whose exception class matches the exception that occurred. Any steps remaining in the try block are skipped. If no exception occurs, then the catch blocks are skipped

If declare a variable within a try block, it will not exist outside the try block, since the curly braces define the scope of the variable. You will often need that variable later, if nowhere else other than the catch or finally blocks, so you would need to declare the variable before the try.

If you declare but don't initialize a variable before a try block, and the only place you set a value for that variable is in the try block, then it is possible when execution leaves the try ... catch structure that the variable never received a value. So, you would get a "possibly uninitialized value" error message from the compiler, since it actually keeps track of that sort of thing.

Usually this happens with object references; you would also generally initialize them to null.

```
public class demo
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
int ans1, ans2;

int a = 2, b = 2, c = 0;
try

{

    ans1 = a/b;

    System.out.println("a/b = " + ans1);
    ans2 = a/c;

    System.out.println("a/c = " + ans2);

}

catch(ArithmeticException e)

{

    System.out.println("Arithmetic Exception!");

}

System.out.println("demo is over");

}
```

```
}
```

Output:

```
C:\>set path=C:\Java\jdk1.5.0_01\bin
```

```
C:\>javac demo.java
```

```
C:\>java demo
```

```
a/b = 1
```

Arithmetic Exception!

```
demo is over
```

Code Explanation –

The program will print the first result, and then not succeed while performing the division for the second equation. Execution will step to the catch block to print our message on the screen

Example -

The prior example used a RuntimeException, which your code is not obligated to handle. Most methods in the I/O classes throw IOException, which is an exception that you must handle.

Following program shows the use of IOException.

```
import java.io.IOException;public class demo
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int num = 0;
```

```
num = System.in.read();

try

{

num = System.in.read(); System.out.println("You entered " + (char)
num);

}

catch (IOException e)

{

System.out.println("IO Exception occurred");

}

}
```

Output:

C:\>javac demo.java

demo.java:11: unreported exception java.io.IOException; must be caught or declared to be thrown

```
num = System.in.read(); // comment out this line
```

^

1 error

Code Explanation:

The line marked to comment out throws `IOException`, but is not in a try block, so the compiler rejects it. The second read attempt is within a try block, as it should be.

- there is no way we can force an `IOException` from the keyboard to test the catch block.

### 7.3.4 Using Multiple catch Blocks

It is possible that a statement might throw more than one kind of exception

- you can list a sequence of catch blocks, one for each possible exception
- remember that there is an object hierarchy for exceptions –

```
class demo
```

```
{
```

```
public static void main (String args [])
```

```
{
```

```
int A[] = new int[5];
```

```
try
```

```
{
```

```
for (int c = 0; c <5; c++)
```

```
{
```

```
//do nothing
```

```
}  
  
for (int c = 0; c <5; c++)  
  
    {  
  
    A[c] = c/ c;  
    }  
  
    }  
  
    catch (ArrayIndexOutOfBoundsException e)  
  
    {  
  
    System.out.println ("Array out of bound ");  
  
    }  
  
    catch (ArithmeticException e)  
  
    {  
  
    System.out.println ("Zero divide error");  
  
    }  
    }  
}
```

Output:

```
C:\>javac demo.java
```

```
C:\>java demo
```

Zero divide error

C:\>

### 7.3.5 finally Block

To guarantee that a line of code runs, whether an exception occurs or not, use a finally block after the try and catch blocks

The code in the finally block will *almost always* execute, even if an unhandled exception occurs; in fact, even if a return statement is encountered

- if an exception causes a catch block to execute, the finally block will be executed after the catch block
- if an uncaught exception occurs, the finally block executes, and then execution exits this method and the exception is thrown to the method that called this method

Syntax –

```
try
```

```
{
```

```
risky code/ unsafe code block
```

```
}
```

```
catch (ExceptionClassName exceptionObjectName)
```

```
{
```

```
code to resolve problem
```

```
}
```

finally

```
{
```

code that will always execute

```
}
```

In summary:

- a try block is followed by zero or more catch blocks
- There may one finally block as the last block in the structure.
- There must be at least one block from the collective set of catch and finally after the try.

It's possible to have a try block followed by a finally block, with no catch block

- this is used to prevent an unchecked exception from exiting the method before cleanup code can be executed

Example:

```
public class demo
```

```
{
```

```
public static void main(String args[])
```

```
{
```

```
try
```

```
{
```

```
System.out.println("Try Block before the error.");
```

```
System.out.println(1/0);
```

```
System.out.println("Try Block after the error.");
}

catch(java.lang.ArithmeticException e)
{

System.out.println("Catch Block");
System.out.println("A Stack Trace of the Error:");
e.printStackTrace();
//e.getMessage();
System.out.println("The operation is not possible.");
}

finally
{
System.out.println("Finally Block");
}
System.out.println("demo is over");
}
}
```

Output:

```
C:\>javac demo.java
```

```
C:\>java demo
```

Try Block before the error.

Catch Block

A Stack Trace of the Error:

```
java.lang.ArithmeticException: / by zero
```

```
    at demo.main(demo.java:8)
```

The operation is not possible.

Finally Block

demo is over

### **7.3.6 Throwing an Exception**

You can throw an exception explicitly using the throw statement. Example:

You need to throw an exception when a user enters a wrong student ID or password.

The throws clause is used to list the types of exception that can be thrown in the execution of a method in a program.

#### **7.3.6.1 Using the throw Statement**

1. The throw statement causes termination of the normal flow of control of the java code and prevents the execution of the subsequent statements.
2. The throw clause convey the control to the nearest catch block handling the type of exception object throws.
3. If no such catch block exists, the program terminates.

The throw statement accepts a single argument, which is an object of the Exception class. Syntax –

```
throw ThrowableObj
```

You can use the following code to throw the `IllegalStateException` exception: class demo

```
{  
  
static void tdemo()  
  
{  
  
try  
  
{  
  
throw new IllegalStateException ();  
  
}  
  
catch (NullPointerException e)  
  
{  
  
System.out.println ("Not Caught by the catch block inside tdemo ().");  
  
}  
  
}  
  
public static void main (String args[ ])  
  
{
```

```
try
{
tdemo();
}
catch(IllegalStateException e)
{
System.out.println("Exception Caught in:"+e);
}
}
}
```

## Output

```
C:\>javac demo.java
```

```
C:\>java demo
```

```
Exception Caught in:java.lang.IllegalStateException
```

```
C:\>
```

### 7.3.6.2 Using the throws Statement

The throws statement is used by a method to specify the types of exceptions the method throws. If a method is capable of raising an exception that it does not handle, the method must specify that the exception have to be handled by the calling method.

This is done using the throws statement. The throws clause lists the types of exceptions that a method might throw. Syntax –

[< access specifier >] [< modifier >] < return type > < method name > [< arg list >] [ throws <exception list >] Example:.

You can use the following code to use the throws statement: class demo

```
{

static void throwMethod () throws ClassNotFoundException

{

System.out.println ("In throwMethod ");
throw new ClassNotFoundException ();

}

public static void main (String args [ ])
{

try

{

throwMethod ();

}

catch ( ClassNotFoundException e)
```

```
{  
  
System.out.println (" throwMethod has thrown an Exception :" +e);  
  
}  
  
}  
  
}
```

## Output

```
C:\>javac demo.java
```

```
C:\>java demo
```

In throwMethod

```
throwMethod has thrown an Exception :java.lang.ClassNotFoundException
```

## 7.3.9 Creating and Using Your Own Exception Classes

You can create your own exception class by extending an existing exception class

Syntax –

```
[modifiers] NewExceptionClassName extends ExceptionClassName
```

```
{  
  
    create constructors that usually delegate to super-constructors  
  
}
```

You could then add any fields or methods that you wish, although often that is not required. You must, however, override any constructors you wish to use: `Exception()`, `Exception(String message)`, `Exception(String message, Throwable cause)`, `Exception(Throwable cause)`. Usually you can just call the equivalent super-constructor. If you extend `RuntimeException` or one of its subclasses, your exception will be treated as a runtime exception.

When a situation arises for which you would like to throw the exception, use the `throw` keyword with a new object from your exception class, for example:

Syntax –

```
throw new ExceptionClassName(messageString);
```

## I/O Packages

### 8.1 Introduction

Stream is an abstract demonstration of input or output device. By using stream, we can write or read data. To bring in information, a program is open a stream on an information source (a file, memory, a socket) and read information sequentially. In this unit, we will learn the concept of stream, I/O package.

### 8.2 Stream:

The Java Input/Output (I/O) is a part of **java.io** package. The `java.io` package contains a relatively large number of classes that support input and output operations. The classes in the package are primarily abstract classes and stream-oriented that define methods and subclasses which allow bytes to be read from and written to files or other input and output sources.

**For reading the stream:**

Open the stream

Read information

Close the stream

### **For writing in stream:**

Open the stream

Write information

Close the stream

There are two types of stream as follows:

- Byte stream
- Character stream

### **8.2.2 Byte Streams:**

It supports 8-bit input and output operations. There are two classes of byte stream

- InputStream
- OutputStream

#### **8.2.2.1 InputStream:**

The **InputStream** class is used for reading the data such as a byte and array of bytes from an input source. An input source can be a **file**, a **string**, or **memory** that may contain the data. It is an abstract class that defines the programming interface for all input streams that are inherited

from it. An input stream is automatically opened when you create it. You can explicitly close a stream with the **close()** method, or let it be closed implicitly when the object is found as a garbage.

The subclasses inherited from the **InputStream** class can be seen in a hierarchy manner shown below:

InputStream

- ByteArrayInputStream
- FileInputStream
- ObjectInputStream
- FilterInputStream
- PipedInputStream
- StringBufferInputStream
- FilterInputStream
  - BufferedInputStream
  - DataInputStream
  - LineNumberInputStream
  - PushbackInputStream

### 8.2.1.2 **OutputStream:**

The `OutputStream` class is a sibling to `InputStream` that is used for writing byte and array of bytes to an output source. Similar to input sources, an output source can be anything such as a file, a string, or memory containing the data. Like an input stream, an output stream is automatically opened when you create it. You can explicitly close an output stream with the `close()` method, or let it be closed implicitly when the object is garbage collected.

The classes inherited from the **OutputStream** class can be seen in a hierarchy structure shown below:

`OutputStream`

- `ByteArrayOutputStream`
- `FileOutputStream`
- `ObjectOutputStream`
- `FilterInputStream`
- `PipedOutputStream`
- `StringBufferInputStream`
- `FilterOutputStream`
  - `BufferedOutputStream`
  - `DataOutputStream`

- PrintStream

OutputStream is also inherited from the Object class. Each class of the OutputStream provided by the java.io package is intended for a different purpose.

### 8.2.2 Character Streams:

It supports 16-bit Unicode character input and output. There are two classes of character stream as follows:

- Reader
- Writer

These classes allow internationalization of Java I/O and also allow text to be stored using international character encoding.

#### 8.2.2.1 Reader:

- BufferedReader
  - LineNumberReader
- CharAraayReader
- PipedReader
- StringReader
- FilterReader
  - PushbackReader

- InputStreamReader

- o FileReader

#### 8.2.2.2 Writer:

- BufferedWriter

- CharArayWriter

- FileWriter

- PipedWriter

- PrintWriter

- String Writer

- OutputStreamWriter

- o FileWriter

### 8.3 How Files and Streams Work:

Java uses **streams** to handle I/O operations through which the data is flowed from one location to another. For example, an **InputStream** can flow the data from a disk file to the internal memory and an **OutputStream** can flow the data from the internal memory to a disk file. The disk-file may be a text file or a binary file. When we work with a text file, we use a **character** stream where one character is treated as per byte on disk. When we work with a binary file, we use a **binary** stream.

## 8.7 Classes:

The following lists of classes are provided by the **java.io** package shown in the table:

| Class                 | Description                                                                                                                        |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------|
| BufferedInputStream   | It used for creating an internal buffer array.<br>It supports the mark and reset methods.                                          |
| BufferedOutputStream  | This class used for writes byte to output stream. It implements a bufferedoutput stream.                                           |
| BufferedReader        | This class provides read text from character input stream and buffering characters. It also reads characters, arrays and lines.    |
| BufferedWriter        | This class provides write text from character output stream and buffering characters. It also writes characters, arrays and lines. |
| ByteArrayInputStream  | It contains the internal buffer and read data from the stream.                                                                     |
| ByteArrayOutputStream | This class used for data is written into byte array. This is implemented in output stream class.                                   |
| CharArrayReader       | It used for char input stream and implements a character buffer.                                                                   |
| CharArrayWriter       | This class also implements a character buffer and it uses an writer.                                                               |
| DataInputStream       | This class reads the primitive data types from the input stream in a machine format.                                               |

|                  |                                                                                      |
|------------------|--------------------------------------------------------------------------------------|
| DataOutputStream | This class writes the primitive data types from the output stream in machine format. |
| File             | This class shows a file and directory pathnames.                                     |
| FileDescriptor   | This class uses for create a FileInputStream and FileOutputStream.                   |
| FileInputStream  | It contains the input byte from a file and implements an input stream.               |
| FileOutputStream | It uses for writing data to a file and also implements an output stream.             |
| FilePermission   | It provides the permission to access a file or directory.                            |
| FileReader       | This class used for reading characters file.                                         |

|                            |                                                                                                                     |
|----------------------------|---------------------------------------------------------------------------------------------------------------------|
| FileWriter                 | This class used for writing characters files.                                                                       |
| InputStream                | This class represents an input stream of bytes.                                                                     |
| InputStreamReader          | It reads bytes and decodes them into characters.                                                                    |
| LineNumberReader           | This class has a line numbers                                                                                       |
| ObjectInputStream          | This class used for recover the object to serialize previously.                                                     |
| ObjectInputStream.GetField | This class access to president fields read from input stream.                                                       |
| ObjectOutputStream         | This class used for writing the primitive data types and also to write the object to read by the ObjectInputStream. |
| ObjectStreamClass          | Serialization's descriptor for classes.                                                                             |

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| ObjectStreamField  | This class describes the serializable field.                                      |
| OutputStream       | This class represents an output stream of bytes.                                  |
| OutputStreamWriter | It writes bytes and decodes them into characters.                                 |
| StringReader       | This is a character string class. It has character read source.                   |
| StringWriter       | This is also a character string class. It uses to shows the output in the buffer. |
| Writer             | It uses for writing to character stream.                                          |

#### 8.5 Exceptions Classes:

The following summary of the exception classes provided by the **java.io** package shown in the table:

| Exceptions              | Description                                                                                                                |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------|
| CharConversionException | It provides detail message in the catch block to associated with the CharConversionException                               |
| EOFException            | This exception indicates the end of file. When the file input stream is to be end then the EOFException is to be occurred. |
| FileNotFoundException   | When the opened file's pathname does not find then this exception occurs.                                                  |

|                          |                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------|
| InterruptedException     | When the I/O operations are interrupted from any causes then it occurs.                       |
| InvalidClassException    | any problems to be created with class, when the Serializing runtime to be detected.           |
| InvalidObjectException   | When the de-serialized objects fails then it occurs.                                          |
| IOException              | When the I/O operations fail then it occurs.                                                  |
| NotActiveException       | The Serialization or deserialization operations are not active then it occurs.                |
| NotSerializableException | This exception occurs when the instance is required to be a Serializable interface.           |
| ObjectStreamException    | This is a supper class of all exception class. It is used for specific Object Stream Classes. |
| WriteAbortedException    | In this exception to be thrown by the ObjectStreamException during a write operating.         |

### 8.6 Standard Streams:

Standard Streams are a feature provided by many operating systems. By default, they read input from the keyboard and write output to the display. They also support I/O operations on files.



**Standard Input:** - Accessed through **System.in** which is used to read input from the keyboard.

- ✓ **Standard Output:** - Accessed through **System.out** which is used to write output to be display.
- ✓ **Standard Error:** - Accessed through **System.err** which is used to write error output to be display.

Java also supports three Standard Streams:

These objects are defined automatically and do not need to be opened explicitly.

Standard Output and Standard Error, both are to write output; having error output separately so that the user may read error messages efficiently.

System.in is a byte stream that has no character stream features. To use Standard Input as a character stream, wrap System.in within the InputStreamReader as an argument.

```
InputStreamReader inp= new InputStreamReader (System.in);
```

## 8.7 Working with Reader classes:

Java provides the standard I/O facilities for reading text from either the file or the keyboard on the command line. The **Reader** class is used for this purpose that is available in the **java.io** package. It acts as an abstract class for reading character streams. The only methods that a subclass must implement are **read(char[], int, int)** and **close()**. The Reader class is further categorized into the subclasses.

The following diagram shows a class-hierarchy of the **java.io.Reader** class.

However, most subclasses override some of the methods in order to provide higher efficiency, additional functionality, or both.

### 8.7.1 InputStreamReader:

An `InputStreamReader` is a bridge from byte streams to character streams i.e. it reads bytes and decodes them into Unicode characters according to a particular platform. Thus, this class reads characters from a byte input stream. When you create an `InputStreamReader`, you specify an `InputStream` from which, the `InputStreamReader` reads the bytes.

The syntax of `InputStreamReader` is written as:

```
InputStreamReader<variable_name>= new InputStreamReader (System.in)
```

### 8.7.2 **BufferedReader:**

The `BufferedReader` class is the subclass of the `Reader` class. It reads character-input stream data from a memory area known as a buffer maintains state. The buffer size may be specified, or the default size may be used that is large enough for text reading purposes.

`BufferedReader` converts an unbuffered stream into a buffered stream using the wrapping expression, where the unbuffered stream object is passed to the constructor for a buffered stream class.

For example the constructors of the `BufferedReader` class shown as:

`BufferedReader (Reader in)` : Creates a buffering character-input stream that uses a default-sized input buffer.

`BufferedReader (Reader in, int sz)`: Creates a buffering character-input stream that uses an input buffer of the specified size.

**BufferedReader** class provides some standard methods to perform specific reading operations shown in the table. All methods throw an `IOException`, if an I/O error occurs.

| Method                              | Return Type | Description                                                           |
|-------------------------------------|-------------|-----------------------------------------------------------------------|
| read( )                             | int         | Reads a single character                                              |
| read(char[] cbuf, int off, int len) | int         | Read characters into a portion of an array.                           |
| readLine( )                         | String      | Read a line of text. A line is considered to be terminated by ('\n'). |
| close( )                            | void        | Closes the opened stream.                                             |

This program illustrates use of standard input stream to read the user input.

```
import java.io.*;

public class ReadStandardIO
{
    public static void main(String[] args) throws IOException
    {
        InputStreamReader inp = new InputStreamReader(System.in)
        BufferedReader br = new BufferedReader(inp);

        System.out.println("Enter text : ");
```

```
String str = in.readLine();

System.out.println("You entered String : ");

System.out.println(str);

}
}
```

### **Output of the Program:**

```
C:\>javac ReadStandardIO.java
C:\>java ReadStandardIO Enter
text:
```

```
this is an Input Stream
You entered String: this
is an Input Stream C:\>
```

The streams provide a simple model for reading and writing data. However, streams don't support all the operations that are common with a disk file. Now, we will learn how to work with a file using the non-stream file I/O.

The `File` class deals with the machine dependent files in a machine-independent manner i.e. it is easier to write platform-independent code that examines and manipulates files using the `File` class. This class is available in the `java.lang` package.

The `java.io.File` is the central class that works with files and directories. The instance of this class represents the name of a file or directory on the host file system.

When a File object is created, the system doesn't check to the existence of a corresponding file/directory. If the files exist, a program can examine its attributes and perform various operations on the file, such as renaming it, deleting it, reading from or writing to it.

The constructors of the File class are shown in the table:

| Constructor          | Description                                                                  |
|----------------------|------------------------------------------------------------------------------|
| File(path)           | Create File object for default directory (usually where program is located). |
| File(dirpath, fname) | Create File object for directory path given as string.                       |

`File (dir, Create File object for directory.name)`

Thus the statement can be written as:

`File f = new File (“<filename>”);`

The methods that are used with the file object to get the attribute of a corresponding file shown in the table.

| Method                  | Description                  |
|-------------------------|------------------------------|
| <code>f.exists()</code> | Returns true if file exists. |

|                                |                                                   |
|--------------------------------|---------------------------------------------------|
| <code>f.isFile()</code>        | Returns true if this is a normal file.            |
| <code>f.isDirectory()</code>   | true if "f" is a directory.                       |
| <code>f.getName()</code>       | Returns name of the file or directory.            |
| <code>f.isHidden()</code>      | Returns true if file is hidden.                   |
| <code>f.lastModified()</code>  | Returns time of last modification.                |
| <code>f.length()</code>        | Returns number of bytes in file.                  |
| <code>f.getPath()</code>       | Path name.                                        |
| <code>f.delete()</code>        | Deletes the file.                                 |
| <code>f.renameTo(f2)</code>    | Renames f to File f2. Returns true if successful. |
| <code>f.createNewFile()</code> | Creates a file and may throw IOException.         |

Whenever the data is needed to be stored, a file is used to store the data. File is a collection of stored information that is arranged in string, rows, columns and lines etc.

Further, we will see how to create a file. This example takes the file name and text data for storing to the file.

For creating a new file `File.createNewFile()` method is used. This method returns a boolean value true if the file is created otherwise return false. If the mentioned file for the specified

directory is already exist then the createNewFile () method returns the false otherwise the method creates the mentioned file and return true.

Let's see an example that checks the existence of a specified file.

```
import java.io.*;
public class CreateFile1
{
    public static void main(String[] args) throws IOException
    {
        File f;
        f=new File
        ("myfile.txt");
        if(!f.exists()){
            f.createNewFile();

            System.out.println("New file \"myfile.txt\" has been created
                                to the current directory");
        }
    }
}
```

First, this program checks, the specified file "**myfile.txt**" is exist or not. If it does not exist then a new file is created with same name to the current location.

#### **Output of the Program**

```
C:\>javac CreateFile1.java
```

```
C:\>java CreateFile1
```

New file "myfile.txt" has been created to the current directory

C:\>

If you try to run this program again then after checking the existence of the file, it will not be created and you will see a message as shown in the output.

C:\>javac CreateFile1.java

C:\>java CreateFile1

the specified file is already exist

C:\>

In Java, it is possible to set dynamic path, which is helpful for mapping local file name with the actual path of the file using the constructing filename path technique.

As seen, how a file is created to the current directory where the program is run. Now we will see how the same program constructs a `File` object from a more complicated file name, using the static constant `File.separator` or `File.separatorChar` to specify the file name in a platform-independent way. If we are using Windows platform then the value of this separator is `'\'`.

Let's see an example to create a file to the specified location.

```
import java.io.*;

public class PathFile

{
    public static void main(String[] args) throws IOException

    {
        File f;
```

```
        f=new File ("example" + File.separator +
        "myfile.txt"); f.createNewFile ();
        System.out.println

("New file \"myfile.txt\" has been created

        to the specified
        location"); System.out.println

("The absolute path of the file is: "
        +f.getAbsolutePath
        ());
    }
}
```

### **Output of the program:**

```
C:\>javac PathFile.java
```

```
C:\>java PathFile
```

```
New file "myfile.txt" has been created to the specified
location the absolute path of the file is:
```

```
C:\Shubh\example\myfile.txt
```

```
C:\>
```

## 8.8 I/O Streams:

Let's now see some I/O streams that are used to perform reading and writing operation in a file.

Java supports the following I/O file streams.

✓  
FileInputStream

✓  
FileOutputStream

### 8.8.1 FileInputStream:

This class is a subclass of InputStream class that reads bytes from a specified file name. The read () method of this class reads a byte or array of bytes from the file. It returns -1 when the end-of-file has been reached. We typically use this class in conjunction with a BufferedInputStream and DataInputStream class to read binary data. To read text data, this class is used with an InputStreamReader and BufferedReader class. This class throws FileNotFoundException, if the specified file is not exist. You can use the constructor of this stream as:

*FileInputStream (File filename);*

### 8.8.2 **FileOutputStream:-**

This class is a subclass of OutputStream that writes data to a specified file name. The write () method of this class writes a byte or array of bytes to the file. We typically use this class in conjunction with a BufferedOutputStream and a DataOutputStream class to write binary data. To write text, we typically use it with a PrintWriter, BufferedWriter and an OutputStreamWriter class. You can use the constructor of this stream as:

*FileOutputStream (File filename);*

### 8.8.3 DataInputStream:-

This class is a type of `FilterInputStream` that allows you to read binary data of Java primitive data types in a portable way. In other words, the `DataInputStream` class is used to read binary Java primitive data types in a machine-independent way. An application uses a `DataOutputStream` to write data that can later be read by a `DataInputStream`. You can use the constructor of this stream as:

```
DataInputStream (FileOutputStream fip);
```

The following program demonstrates how contents are read from a file.

```
import java.io.*;

public class ReadFile
{
    public static void main(String[] args) throws IOException
    {
        File f;

        f=new File("myfile.txt");
        if(!f.exists()&&
        f.length()<0)

        System.out.println("The specified file is not exist");

        else{
```

```
    FileInputStream finp=new
    FileInputStream(f); byte b;

do{

    b=(byte)finp.read();

    System.out.print((char)b);

}

while(b!=-
    1);
    finp.close();

}

}
```

### **Output of the Program:**

```
C:\>javac ReadFile.java
```

```
C:\>java
```

```
ReadFile this is a
```

```
text file? C:\>
```

This program reads the bytes from file and displays it to the user.

Now we will learn how to write data to a file. As discussed, the `FileOutputStream` class is used to write data to a file.

Let's see an example that writes the data to a file converting into the bytes.

This program first checks the existence of the specified file. If the file exists, the data is written to the file through the object

of the `FileOutputStream` class.

```
import java.io.*;

public class WriteFile

{

    public static void main(String[] args) throws IOException

    {

        File f=new File ("textfile1.txt");
        FileOutputStream fop=new FileOutputStream (f);

        if (f.exists ())

        {

            String str="This data is written through the
                program"; fop.write (str.getBytes ());

            fop.flush
                ();
```

```
fop.close
();
System.out.println ("The data has been written");

}
else
    System.out.println ("This file is not exist");

}
```

### **Output of the Program**

C:\>javac WriteFile.java

C:\>java WriteFile

The data has been written

C:\>

Now, you will learn how to count the availability of text lines in the particular file. A file is read before counting lines of a particular file. File is a collection of stored information that is arranged in string, rows, columns and lines etc. Try it for getting the lines through the following program

### **Description of program:**

The following program helps you in counting lines of a particular file. At the execution time of this program, it takes a file name with its extension from a particular directory and checks it using exists () method. If the file exists, it will count lines of a particular file otherwise it will display a message

—**File does not exists!!**

## Description of code:



### **FileReader (File file):**

This is the constructor of **FileReader** class that is reliable for reading a character files. It constructs a new **FileReader** and takes a file name that have to be read.



### **LineNumberReader ():**

This is the constructor of **LineNumberReader** class. It constructs a new line-numbering reader. It reads characters and puts into buffer. By default the numbering of line begins from '0'.

## Here is the code of program:

```
import java.io.*;

public class NumberOfLine{

public static void main(String[] args)
    { try{

        System.out.println("Getting line number of a particular
file example!");

        BufferedReader bf = new BufferedReader(new
InputStreamReader (System.in));
```

```
System.out.println("Please enter file name with extension:")

;

String str =
bf.readLine(); File file =
new File(str); if
(file.exists()){

    FileReader fr = new FileReader(file);
LineNumberReader ln = new
LineNumberReader(fr); int count = 0;

    while (ln.readLine() !=
        null){ count++;
}

System.out.println("Total line no: " +
count); ln.close();
}

else{

    System.out.println("File does not exists!");
}

}

catch(IOException e){

    e.printStackTrace();
```

```
    }  
  
    }  
}
```

### **Output of program:**

Getting line number of a particular file example!

Please enter file name with extension:

AddTwoBigNumbers.shtml

Total line no: 58

Java provides the facility for changing a file timestamp according to the user reliability.

### **Description of program:**

This program helps you in changing a file timestamp or modification time in Java. After running this program it will take a file name and its modification date in '**dd-mm-yyyy**' format. Then it will check the given file is exist or not using **exists ()** method. When the file exists, this program will change the date of given file and it will display a message "**Modification is successfully!**" otherwise it will show —**File does not exists!**||

### **Description of code:**

✓ **setLastModified(long time):**

This is the method that sets the last modification time of a file or directory and returns Boolean types values either '**true**' or '**false**'. If it will return a 'true' only when the

modification is completely successfully otherwise, it will return 'false'. This method takes following long type data:

✓ **time:**

This is the time that has to be modified or set.

✓ **getTime ():**

This is the method that returns the number of milliseconds in GMT format like: 23-04-2007.

**Here is the code of program:**

```
import java.io.*;
import
java.util.*;

import java.text.*;

public class ChangeFileDate{

    public static void main(String[] args) {
```

```
try{

    System.out.println("Change file timestamp example!");
    BufferedReader bf = new BufferedReader(new InputStreamReader
(System.in));
    System.out.println("Enter file name with extension:");

    String str = bf.readLine();

    System.out.println("Enter last modified date in 'dd-
mm-yyyy' format:");

    String strDate = bf.readLine();

    SimpleDateFormat sdf= new SimpleDateFormat("dd-MM-yyyy");

    Date date = sdf.parse(strDate);

    File file = new File(str);

    if (file.exists()){

        file.setLastModified(date.getTime());

        System.out.println("Modification is successfully!");

    }

    else{

        System.out.println("File does not exists!");

    }

}
```

```
}  
  
    catch(Exception e){  
  
        e.printStackTrace();  
  
    }  
}
```

### **Output of program:**

Change file timestamp  
example! Enter file name  
with extension:  
StrStartWith.shtml

Enter last modified date in 'dd-mm-yyyy'  
format: 23-04-2007

Modification is successfully

### **8.9 Finding a File:-**

To find a file or directory it is very necessary to know the path of the file or directory so that you can access it. If you know the path then it is very easy to work on it. Suppose a situation where a problem comes in front you where you don't know the path of the file, then what will you do?

This problem can be solved by using a method `getAbsolutePath ()`. The method `getAbsolutePath ()` should be used where we don't know the exact path of the file.

To find an absolute path of a file, Firstly we have to make a class **GetAbsolutePath**. Inside this class, define the main method. Inside this method define a File class of java.io package. Inside the constructor of a File class pass the name of the file whose absolute path you want to know. Now call the method *getAbsolutePath ()* of the **File** class by the reference of File class and store it in a String variable. Now print the string, you will get an absolute path of the file.

In this class we have make use of the following things by which this problem can be solved.

- ✓ **File:** It is class in java.io package. It implements Comparable and Serializable interface.
- ✓ **getAbsolutePath ():** It returns the absolute path name in the form of string.

**Code of the program is given below:**

```
import java.io.*;

public class GetAbsolutePath

{

    public static void main(String[] args)

    {

        String str = args[0];

        File file = new File(str);
        String absolutePathOfFirstFile = file.getAbsolutePath();
```

```

System.out.println(" The absolute path in first form is "
                    + absolutePathOfFirstFile);

file = new File( "Happy" + File.separatorChar+ str);
String absolutePathOfSecondFile =
file.getAbsolutePath();

System.out.println(" The absolute path is " +
absolutePathOfSecondFile);

file = new File("Happy" + File.separator + ".." +
File.separator + str);

String absolutePathOfThirdFile = file.getAbsolutePath
(); System.out.println
(" The absolute path is" + absolutePathOfThirdFile);
}

```

### **Output of the program**

Happy

The absolute path in first form is C:\Smile\Happy

The absolute path is C:\Smile\Happy\Happy

The absolute path is C:\Smile\Happy\..\Happy

## Multi threading

**9.0 Objective:** In this lesson of Java Tutorial, you will learn...

- Thread life cycle
- How to create thread
- Advantages of threading

### 9.1 Introduction:

A *thread* is defined as a separate stream of implementation that takes place simultaneously with and independently of everything else that might be happening. It does not have an event loop. A thread runs autonomously of anything else happening in the computer. With threads the other tasks that don't get stuck in the loop can continue processing without waiting for the stuck task to terminate. A thread is a coding that doesn't affect the architecture of an application. Threading is equally separate the computer's power among different tasks.

### 9.2 Overview:

Threading concept is very important in Java Programming language. A thread is a sequential path of code execution within a program. And each thread has its own local variables, program counter and lifetime. In Java, an object of the Thread class can represent a thread. Thread can be implemented through any one of two ways:

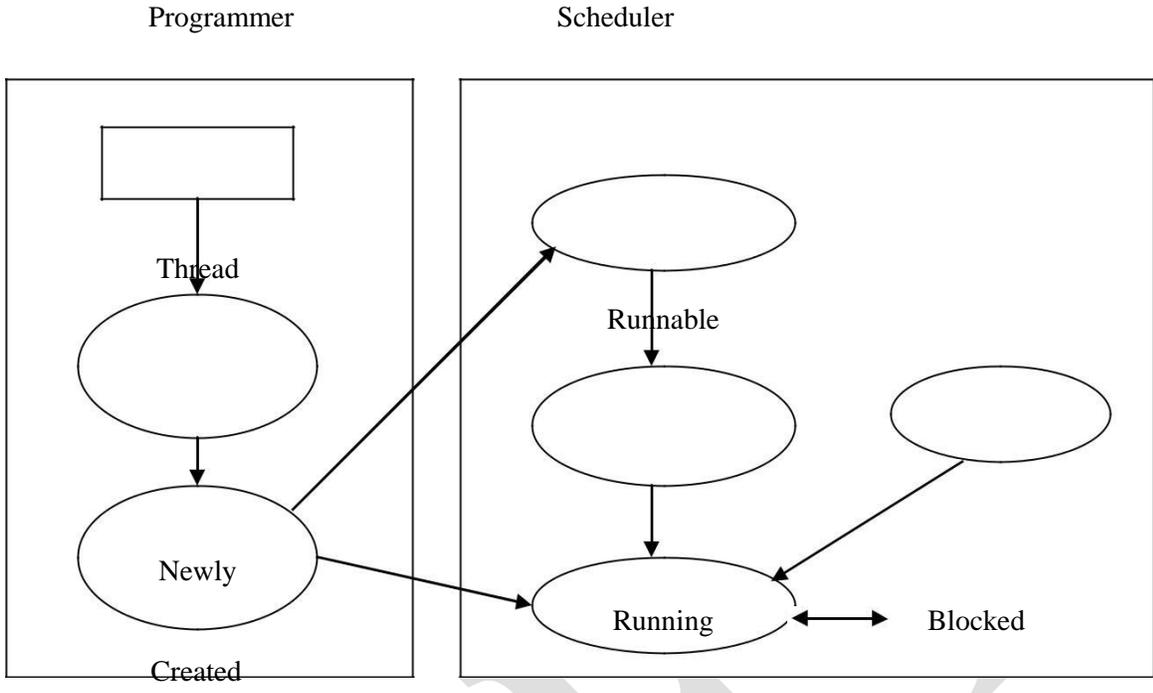
Using threads in Java will enable greater flexibility to programmers looking for that extra edge in their programs. The simplicity of creating, configuring and running threads lets Java programmers devise portable and powerful applets/applications that cannot be made in other third-generation languages. Threads allow any program to perform multiple tasks at once. In an Internet-aware language such as Java, this is a very important tool.

### **9.3.1 Thread Life cycle:**

When you are programming with threads, understanding the life cycle of thread is very valuable.

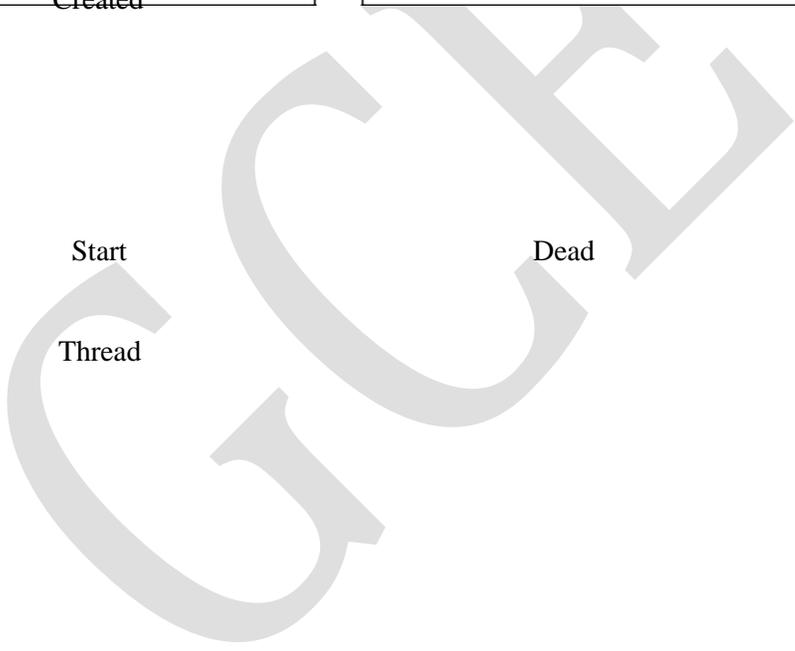
While a thread is alive, it is in one of several states. By invoking start() method, it doesn't mean that the thread has access to CPU and start executing straight away. Several factors determine how it will proceed.

Different states of a thread are:



Start  
Thread

Dead



### **Fig 9.1: Thread Life cycle**

1. New state – After the construction of Thread instance the thread is in this state but before the start() method invocation. At this point, the thread is considered not alive.
2. Runnable (Ready-to-run) state – A thread start its life from Runnable state. A thread first enters runnable state after the invoking of start() method but a thread can come again to this state after either running, waiting, sleeping or coming back from blocked state also. On this state a thread is waiting for a turn on the processor.
3. Running state – A thread is in running state that means the thread is presently executing. There are numerous ways to enter in Runnable state but there is only one way to enter in Running state: the scheduler select a thread from runnable pool.
4. Dead state – A thread can be considered dead when its run() method completes. If any thread comes on this state that means it cannot ever run again.
5. Blocked - A thread can enter in this state because of waiting the resources that are hold by another thread.

#### **9.3.2 Advantages of multithreading over multi-tasking:**

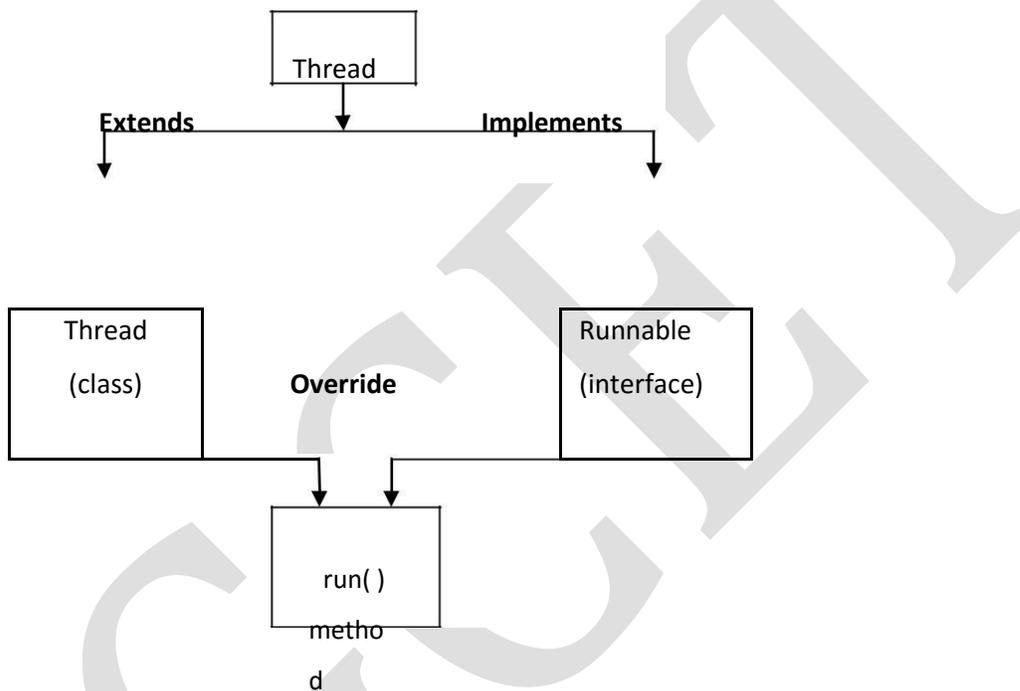
1. Reduces the computation time.
2. Improves performance of an application.
3. Threads distribute the same address space so it saves the memory.
4. Context switching between threads is usually less costly than between processes.

5. Cost of communication between threads is comparatively low.

### 9.3.3 Thread Creation and simple programs:

In Java, an object of the Thread class can represent a thread. Thread can be implemented through any one of two ways:

- Extending the java.lang.Thread Class
- Implementing the java.lang.Runnable Interface



**Fig 9.3: Creation of thread**

- Extending the java.lang.Thread Class

Syntax: class MyThread extends Thread

{

```
}
```

- Implementing the java.lang.Runnable Interface

Syntax: MyThread implements Runnable

```
{
```

```
}
```

- After declaration of thread class, we have to override run( ) method in class.
- 

Now we can create object of thread if needed.

In short we have to follow following these steps:

1. Extend the java.lang.Thread Class.
2. Override the run( ) method in the subclass from the Thread class to define the code executed by the thread.
3. Create an instance of this subclass. This subclass may call a Thread class constructor by subclass constructor.
4. Invoke the start( ) method on the instance of the class to make the thread eligible for running.

The following program demonstrates a single thread creation extending the "Thread" Class:

class MyThread extends Thread

```
{
```

```
String s=null;
```

```
MyThread(String s1)
```

```
{
```

```
        s=s1;

        start();

    }

    public void run()

    {

        System.out.println(s);

    }

}

public class RunThread

{

    public static void main(String args[])

    {

        MyThread m1=new MyThread("Thread started...");

    }

}

}
```

Output of the Program is :

C:\>javac RunThread.java

```
C:\>java RunThread
```

Thread started....

## II. Implementing the java.lang.Runnable Interface

The procedure for creating threads by implementing the Runnable Interface is as follows:

1. A Class implements the Runnable Interface, override the run() method to define the code executed by thread. An object of this class is Runnable Object.
2. Create an object of Thread Class by passing a Runnable object as argument.
3. Invoke the start() method on the instance of the Thread class.

The following program demonstrates the thread creation implenting the Runnable interface:

```
class Thr1 implements Runnable{
    Thread t;

    String s=null;

    Thr1 (String
        s1) { s=s1;

        t=new Thread(this);
        t.start();
    }

    public void run() {
        System.out.println(s
            );
    }
}
```

```
public class RunnableThread{

    public static void main(String args[]){
        Thr1 m1=new Thr1("Thread started....");
    }

}
```

Output:

```
C:\>javac RunnableThread.java
```

```
C:\>java RunnableThread
```

```
Thread started....
```

However, this program returns the output same as of the output generated through the previous program.

There are two reasons for implementing a Runnable interface preferable to extending the Thread Class:

1. If you extend the Thread Class, that means that subclass cannot extend any other Class, but if you implement Runnable interface then you can do this.
2. The class implementing the Runnable interface can avoid the full overhead of Thread class which can be excessive.

### **join() & isAlive() methods:**

The following program demonstrates the join() & isAlive() methods:

```
class DemoAlive extends Thread {
    int value;
```

```
public DemoAlive(String str)
{
    super(str);
    value=0;

    start();
}
public void run()
{
    try
    {
        while (value < 5) {

            System.out.println(getName() + ": " + (value++));
            Thread.sleep(250);

        }
    } catch (Exception e) {}
    System.out.println("Exit from thread: " + getName());
}
}
public class DemoJoin
{
    public static void main(String[] args)
    {

        DemoAlive da = new DemoAlive("Thread a");
        DemoAlive db = new DemoAlive("Thread b");
        try
        {
```

```
        System.out.println("Wait for the child threads to finish.");
        da.join();

        if (!da.isAlive()) System.out.println("Thread
            A not alive.");

        db.join();

        if (!db.isAlive()) System.out.println("Thread
            B not alive.");

    } catch (Exception e) { }
    System.out.println("Exit from Main Thread.");
}
}
```

Output:

```
C:\>javac DemoJoin.java
```

```
C:\>java DemoJoin
```

```
Wait for the child threads to finish.
```

```
Thread a: 0
```

```
Thread b: 0
```

```
Thread a: 1
```

```
Thread b: 1
```

```
Thread a: 2
```

```
Thread b: 2
```

```
Thread a: 3
```

```
Thread b: 3
```

```
Thread a: 4
```

```
Thread b: 4
```

```
Exit from thread: Thread a
```

Thread A not alive.

Exit from thread: Thread b

Thread B not alive.

Exit from Main Thread.

---

### 9.3.4 Synchronized threads:

In Java, the threads are executed separately to each other. These types of threads are called as asynchronous threads. But there are two problems may be occurs with asynchronous threads.

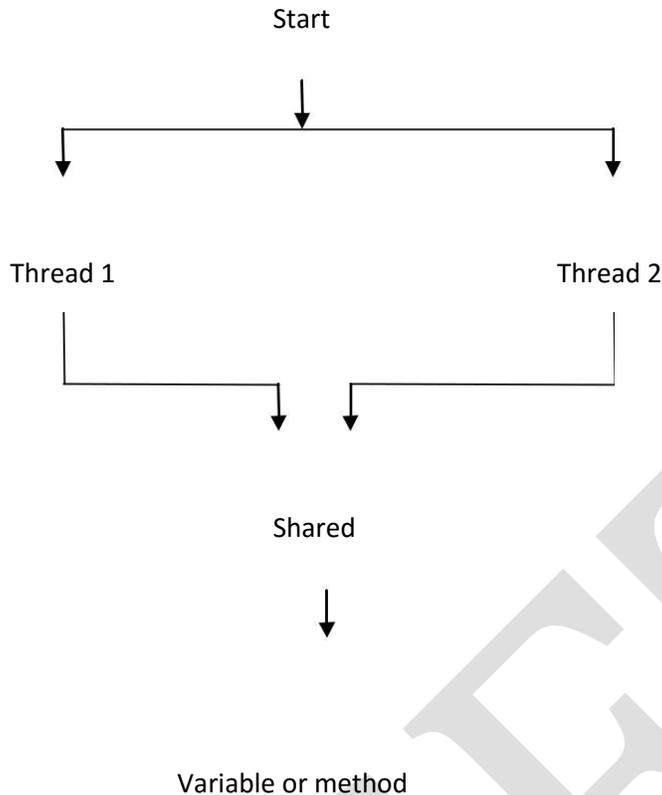
- Two or more threads share the similar resource (variable or method) while only one of them can access the resource at one time.
- If the producer and the consumer are sharing the same kind of data in a program then either producer may make the data faster or consumer may retrieve an order of data and process it without its existing.

Suppose, we have created two methods as `increment()` and `decrement()`. which increases or decreases value of the variable "count" by 1 respectively shown as:

```
public void increment() {  
    count++; }  
}
```

When the two threads are executed to access these methods (one for `increment()`, another for `decrement()`) then both will distribute the variable "count". in that case, we can't be sure that what value will be returned of variable "count".

We can see this problem in the diagram shown below:



To avoid this problem, Java uses monitor also known as —semaphore to prevent data from being corrupted by multiple threads by a keyword synchronized to coordinate them and intercommunicate to each other. It is basically a mechanism which allows two or more threads to share all the available resources in a sequential manner. Java's synchronized is used to ensure that only one thread is in a critical region. Critical region is a lock area where only one thread is run (or lock) at a time. Once the thread is in its critical section, no other thread can enter to that critical region. In that case, another thread will has to wait until the current thread leaves its critical section.

General form of the synchronized statement is as:

```

synchronized(object) {

// statements to be synchronized

}
  
```

- 

## **Lock:**

Lock term refers to the access approved to a particular thread that can access the shared resources. At any given time, only one thread can hold the lock and thereby have access to the shared resource. Every object in Java has build-in lock that only comes in action when the object has synchronized method code. By associating a shared resource with a Java object and its lock, the object can act as a guard, ensuring synchronized access to the resource. Only one thread at a time can access the shared resource guarded by the object lock.

Since there is one lock per object, if one thread has acquired the lock, no other thread can acquire the lock until the lock is not released by first thread. Acquire the lock means the thread currently in synchronized method and released the lock means exits the synchronized method.

Remember the following points related to lock and synchronization:

Only methods (or blocks) can be synchronized, Classes and variable cannot be synchronized.

- Each object has just one lock.
- All methods in a class need not to be coordinated. A class can have both synchronized and non-synchronized methods.
- If two threads wants to execute a synchronized method in a class, and both threads are using the similar instance of the class to invoke the method then only one thread can execute the method at a time.
- If a class has both synchronized and non-synchronized methods, multiple threads can still access the class's non-synchronized methods. If you have methods that don't access the data you're trying to protect, then you don't need to synchronize them. Synchronization can cause a hit in several cases (or even deadlock if used incorrectly), so you should be careful not to overuse it.
- If a thread goes to sleep, it holds any locks it has—it doesn't let go them.
-

A thread can obtain more than one lock. For example, a thread can enter a synchronized method, thus acquiring a lock, and then directly invoke a synchronized method on a different object, thus acquiring that lock as well. As the stack unwinds, locks are unrestricted again.

- You can synchronize a block of code rather than a method.
- Constructors cannot be synchronized

#### 9.3.4.1 Synchronized Methods:

Any method is specified with the keyword `synchronized` is only executed by one thread at a time. If any thread wants to implement the synchronized method, firstly it has to obtain the objects lock. If the lock is already held by another thread, then calling thread has to wait. Synchronized methods are useful in those situations where methods are executed concurrently, so that these can be intercommunicate control the state of an object in ways that can corrupt the state if . Stack implementations usually define the two operations push and pop of elements as synchronized, that's why pushing and popping are mutually exclusive process. For Example - if several threads were sharing a stack, if one thread is popping the element on the stack then another thread would not be able to pushing the element on the stack.

The following program demonstrates the synchronized method:

```
class Demo extends Thread{

    static String msg[]={"This", "is", "a", "synchronized",
"variable"}; Share(String threadname){
    super(threadname);
}

    public void run(){
        display(getName()
);
    }
```

```

    }

    public synchronized void display(String
    threadN){ for(int i=0;i<=4;i++)
        System.out.println(threadN+msg[i]);
        try{

            this.sleep(1000);
        }catch(Exception e){}

    }
}
public class SynThread1 {

    public static void main(String[] args)    {
        Share t1=new Share("Thread One: ");

        t1.start();
        Share t2=new Share("Thread Two: ");
        t2.start();

    }
}

```

Output of the program is:

Thread One: variable

Thread Two: This

Thread Two: is

Thread two: a

Thread Two: synchronized

Thread Two: variable

C:\nisha>javac SynThread.java

```
C:\nisha>java SynThread
```

```
Thread One: This
```

```
Thread One: is
```

```
Thread One: a
```

```
Thread One: synchronized
```

```
Thread One: variable
```

```
Thread Two: This
```

```
Thread Two: is
```

```
Thread two: a
```

```
Thread Two: synchronized
```

```
Thread Two: variable
```

## ***9.4 Summary:***

A thread executes a series of instructions. Every line of code that is executed is done so by a thread. In Java, the threads are executed independently to each other. Multithreading is vital to Java for two main reasons. First, multithreading enables you to write very efficient programs because it lets you utilize the idle time that is present in most programs. Most I/O devices, whether they be network ports, disk drives, or the keyboard, are much slower than the CPU. Thus, a program will often use a majority of its execution time waiting to send or receive information to or from a device. By using multithreading, your program can execute another task during this idle time. For example, while one part of your program is sending a file over the Internet, another part can be

handling user interaction (such as mouse clicks or button presses), and still another can be buffering the next block of data to send.

The second reason that multithreading is important to Java relates to Java's eventhandling model.

A program (such as an applet) must respond speedily to an event and then return. An event handler must not retain control of the CPU for an extended period of time.

10

## Applets

### 10.1 Introduction to Applet

There are two kinds of Java programs, applications (also called stand-alone programs) and Applets. An **Applet** is a small Internet-based program that has the Graphical User Interface (GUI), written in the Java programming language.

Applets are designed to run inside a web browser or in applet viewer to facilitate the user to animate the graphics, play sound, and design the GUI components such as text box, button, and radio button. When applet arrives on the client, it has limited access to resources, so that it can produce arbitrary multimedia user interface and run complex computation without introducing the risk of viruses or breaching data integrity.

To create an applet, we extend the `—java.applet.Applet` class And by overriding the methods of `java.awt.Applet`, new functionality can be placed into web pages.

Applets are compiled using javac compiler and it can be executed by using an appletviewer or by embedding the class file in the HTML (Hyper Text Markup Language) file.

## 10.2 Applet vs Application

- Applets as previously described, are the small programs while applications are larger programs.
- Applets don't have the main method while in an application execution starts with the main method.
- Applets are designed just for handling the client site problems. while the java applications are designed to work with the client as well as server.
- Applications are designed to exist in a secure area. while the applets are typically used.
- Applications are not too small to embed into a html page so that the user can view the application in your browser. On the other hand applet have the accessibility criteria of the resources.

## 10.3 Applet class

The **java.applet** package is the smallest package in Java API(Application Programming Interface). The **Applet class** is the only class in the package. The Applet class has many methods that are used to display images, play audio files etc but it has no main() method. Some of them were explained below that give you the knowledge about Applets and their behavior.

**init()** : This method is used for whatever initializations are needed for your applet. Applets can have a default constructor, but it is better to perform all initializations in the init method instead of the default constructor.

**start()** : This method is automatically called after Java calls the init method. If this method is overwritten, code that needs to be executed every time that the user visits the browser page that contains this applet.

**stop()** : This method is automatically called when the user moves off the page where the applet sits. If your applet doesn't perform animation, play audio files, or perform calculations in a thread, you don't usually need to use this method.

**destroy():** Java calls this method when the browser shuts down.

#### **10.4 Advantages of Applet**

Following are the advantages of a Java Applet:

- The most important feature of an Applet is, It is truly platform independent so there is no need of making any changes in the code for different platform i.e. it is simple to make it work on Linux, Windows and Mac OS i.e. to make it cross platform.
- The same applet can work on "all" installed versions of Java at the same time, rather than just the latest plug-in version only.
- It can move the work from the server to the client, making a web solution more scalable with the number of users/clients.
- The applet naturally supports the changing user state like figure positions on the chessboard.
-

Applets improves with use: after a first applet is run, the JVM is already running and starts quickly.

- Applets can be used to provide dynamic user-interfaces and a variety of graphical effects for web pages.

## 10.5 Applet Lifecycle

Every java Applet inherits a set of default behaviours from the Applet class. As a result, when an applet is loaded it undergoes a series of changes in its state. Following are the states in applets lifecycle.

### 1) **Born or Initialisation state :**

An applet begins its life when the web browser loads its classes and calls its `init()` method. This method is called exactly once in Applets lifecycle and is used to read applet parameters. Thus, in the `init()` method one should provide initialization code such as the initialization of variables.

Eg. 

```
public void init()
{
    //initialisation
}
```

### 2) **Running State:**

Once the initialization is complete, the web browser will call the `start()` method in the applet. This method must called atleast once in the Applets lifecycle as the `start()` method can also

be called if the Applet is in —Stopped state. At this point the user can begin interacting with the applet.

```
Eg.  public void start()
      {
          //Code
      }
```

### 3) Stopped State:

The web browser will call the Applets stop() method, if the user moved to another web page while the applet was executing. So that the applet can take a breather while the user goes off and explores the web some more. The stop() method is called atleast once in Applets Lifecycle.

```
Eg.  public void stop()
      {
          //Code
      }
```

### 4) Dead State:

Finally, if the user decides to quit the web browser, the web browser will free up system resources by killing the applet before it closes. To do so, it will call the applets destroy() method. One can override destroy() to perform one-time tasks upon program completion. for example, cleaning up threads which were started in the init() method.

```
Eg.  public void destroy()
      {
          // Code
```

```
}
```

**Note:** If the user returns to the applet, the web browser will simply call the applet's start() method

again and the user will be back into the program.

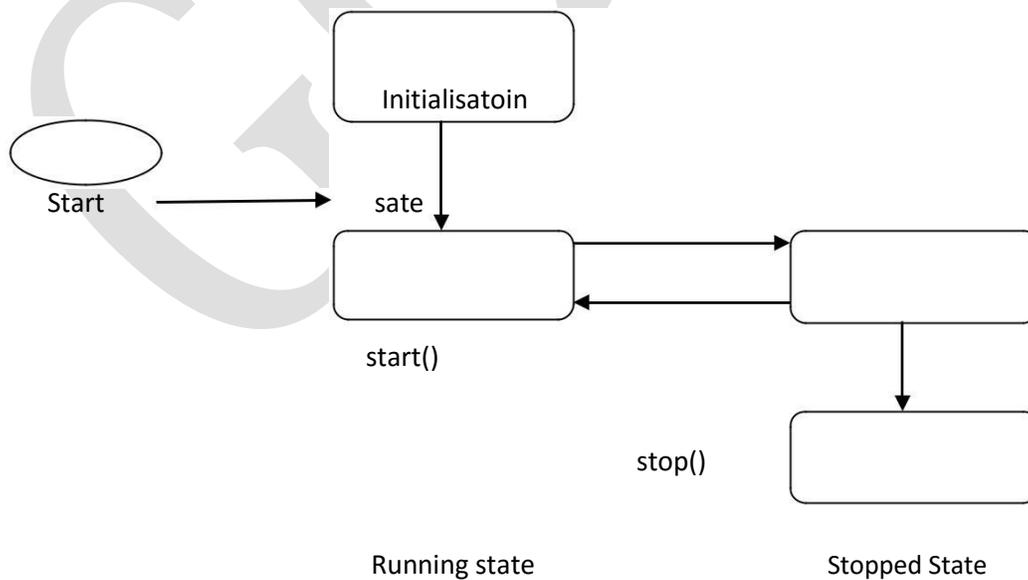
### 5) Display State :

Applet moves to the display state whenever it has to perform the output operations on the screen. This happens immediately after the applet enters into the running state. The paint() method is called to accomplish this task.

Eg. 

```
public void paint(Graphics g)
{
    //Display Statements
}
```

One can show Lifecycle of an Applet Graphically as follows:



start()

destroy()

Dead State

## 10.6 My First Applet

The following example is made simple enough to illustrate the essential use of Java applets through its java.applet package.

### Example.

```
import java.awt.*; import
java.applet.*;

public class SimpleApplet extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("My First Applet",40,40);
    }
}
```

- 
- 

Save the file as **SimpleApplet.java**

- Compile the file using **javac SimpleApplet.java**

Here is the illustration of the above example,

In the first line we import the Abstract Window Toolkit(AWT) classes as Applet interact with the user through the AWT, not through the console –based I/O classes. The AWT contains support for a window based graphical interface.

- In the second line we import the Applet package, which contains the class —Applet. As every applet that we create is the subclass of Applet.
- The next line declares the class SimpleApplet. This class must be declared in public, because it will be accessed by code that is outside the program.
- Inside simpleApplet, paint() method is declared. This method is defined by the AWT and must be overridden by the Applet. Method paint() is called each time that the applet must redisplay its output.

This paint() method has parameter of type — Graphics. This parameter contains the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

- Inside paint() method is a call to drawstring(), which is a member of the Graphics class. This method output a String beginning at specified X, Y locations.

### **How to run an Applet?**

- There are two ways in which one can run an applet, as follows

- 1) Executing the applet within a java-compatible web browser.
- 2) Using an applet viewer, such as the standard SDK tool, `—appletviewer`. An applet viewer executes your applet in a window. This is generally the fastest and easiest way to test your applet.

- 

To execute an applet in a web browser, you need to write a short HTML text file that contains the appropriate APPLET tag.

For above example it is

```
<html>
```

```
<body>
```

```
<applet code="SimpleApplet.class" width=200 height=100>
```

```
</applet>
```

```
</body>
```

```
</html>
```

- Save this code in text file with extension **.html** say **Myapplet.html**.

- 

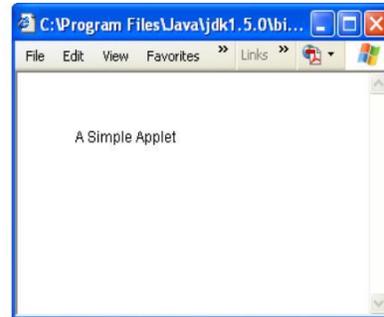
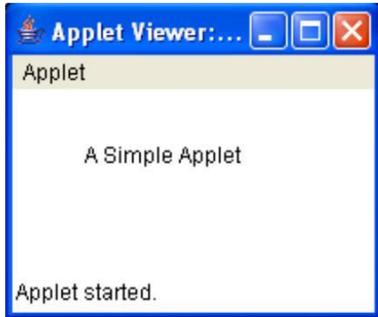
- 

- Compile the file using **javac SimpleApplet.java**

On successful compilation of SimpleApplet.java file, execute the this file using **appletviewer Myapplet.html** or just open this html file directly.

The output of above example appears as shown in the following figure :

OR



Insted of creating different text file for html code one can write above program as follows

```
import java.awt.*; import
java.applet.*;

/* <applet code="SimpleApplet" width=200 height=100>
</applet>

*/

public class SimpleApplet extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("My First Applet",40,40);
    }
}
```

•  
•  
•

- 

Save the file as **SimpleApplet.java**

Compile the file using **javac SimpleApplet.java**

On successful compilation, execute the this file using **appletviewer SimpleApplet.java**

The output remains same.

### **Building an applet code:**

- Applet code uses the series of two classes, namely Applet and Graphics from java class library.

Applet class which is contained in the java.applet package provides life and behaviour to the applet through its methods such as `init()`, `start()`, and `paint()`.

- When an applet is loaded, java automatically calls a series of applet class methods for starting, running and stopping the applet code.
- The applet class therefore maintains the lifecycle of an applet.
- The `paint()` method of the applet class, when it is called, actually display the result of applet code on the screen.
- The output may be text, graphics or sound.
- The `paint()` method, which requires a Graphics object as an argument, is defined as follows:

```
public void paint(Graphics g)
```

-

This requires that the applet code imports the java.awt package that contains the Graphics class.

- All output operations of an applet are performed using the methods defined in the Graphics class.

## 10.7 Applet tag

The Applet tag is used to start an applet from both HTML document and form applet viewer.

An applet viewer will execute each Applet tag that it finds in a separate window, while web browsers like Netscape Navigator, Internet Explorer and HotJava will allow many applets in a single page.

The <applet....> tag included in the body section of HTML file supplies the name of the applet to be loaded and tells the browser how much space the applet requires

The syntax for the standard Applet tag is as follows

```
<applet[codebase=codebaseURL] code=|Applet file|
```

```
  [ALT=|alternative text|
```

```
  [name=AppletInstanceName]
```

```
  Width=pixels height= pixels [align=
  alignment]
```

```
>
```

```
[<param name=|Attributename| value =|Attribute value|] [<param
name=|Attributename| value =|Attribute value|]
```

```
.....
```

[HTML displayed in the absence of java]

</applet>

Here is meaning of each peice of above code

- **Codebase:** Codebase is an optional attribute that specifies the base URL of the applet code, which is the directory that will be searched for te applet's executable class file. The HTML document's URL directory is used as the CODEBASE if this attribute is not specified. The CODEBASE if this attribute is not specified. The CODEBASE does not have to be on the host from which the HTML document was read.
- **Code:** code is required attribute that gives the name of the file containing the applets compiled .class file. This file is relative t the code base URL of the applet , which is the directory that the HTML file whs in or th edirectory indicated by the CODEBASE if set.
- **ALT :** The ALT tag is an optional attribute used to specify a short text message that should be displayed if browser understand the APPLETTAG tag but cant currently run java applet.
- **Name:** Name is an optional attribute used to specify a name for the applet instance. Applets must be named in order for other applets on the same page to find them by name and communicate with them. To obtain an applet by name, use getAppet(), which is defined by the AppletContext interface.
- **Param name and value :** The PARAM tag allows us to specify applet specific arguments in an HTML page. Applets access their attributes with the getParameter() method.

## 10.8 Passing Parameters to Applet

One can supply user-defined parameters to an applet using <param.....> tag. Each <param.....> tag has a **name** attribute such as color,and a **value** attribute such as red. Inside the applet code, the

applet can refer to that parameter by name to find its value. For e.g. the color of the text can be changed to red by an applet using a <param...> tag as follows

```
<applet....>  
    <param=color value = —red||>  
</applet>
```

Similarly we can change the text to be displayed by an applet by supplying new text to the applet through a <param...>tag as shown below.

```
<param name=text value = —xyzl >
```

Passing a parameters to an applet is similar to passing parameters to main() method using command line arguments. To set up and handle parameters, we need to do two things.

- 1) Include appropriate <param.....> tags in the HTML document.
- 2) Provide code in the applet to pass these paraments.

Parameters are passed to an applet when it is loaded. We can define the init() method in the applet to get hold of the parameters defined in the <param> tags. This is done using the getparameter() method, which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

## 10.9 Types of Applets

As we can embed applet into web pages in two ways i.e. by writting our own applet and then embed into web pages. Or by downloading it from a remote computer system and then embed it into webpage.

An applet developed locally and stored in a local system is known as **local applet**. Therefore when webpage is trying to find local applet it does not need the internet connection.

A **remote applet** is that which is developed by someone else and stored on a remote computer connected to the internet. If our system is connected to the internet then we can download it from the remote computer and run it. In order to locate and load a remote applet, we must know the applet's address on the web. This address is known as Uniform Resource Locator (URL) and must be specified in the applet's document.

## 10.10 Examples

**Example 1** // Example to illustrate Applet Lifecycle

```
import java.awt.*; import
java.applet.*;

/* <applet code="AppletTest" width=200 height= 100>
</applet>

*/

public class AppletTest extends Applet

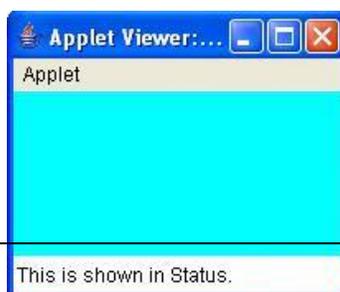
{
    public void init()
    {
        System.out.println("Applet Initialised...");
        setBackground(Color.cyan);
    }
}
```

```
public void start()
{
    System.out.println("Applet Started....");
}
public void stop()
{
    System.out.println("Applet Stoppen....");
}
public void destroy()
{
    System.out.println("Applet Destroyed....");
}
public void paint(Graphics g)
{
    g.drawString("Applet Text",200,400); showStatus("This
    is shown in Status.");
}
}
```

- Save the file as **AppletTest. Java**
- 
- Compile the file using **javac AppletTest.java**

On successful compilation, execute the file using **appletviewer AppletTest.java**

The output appers as shown in following figure :



**Example 2** // Example to illustrate Applet Lifecycle

```
import java.awt.*; import
java.applet.*;

/* <applet code="Sample" width=200 height= 100> </applet>

*/

public class Sample extends Applet
{
    String msg;

    public void init()
    {
        setBackground(Color.cyan);

        setForeground(Color.red); msg = "Inside init()-
        ";
    }

    public void start()
    {
        msg += "Inside start()-";
    }
}
```

```
public void paint(Graphics g)
{

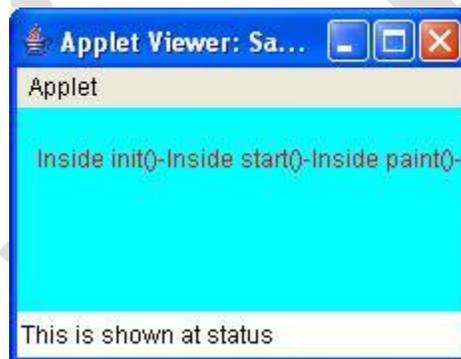
    msg += "Inside paint()-"; g.drawString(msg,10,30);
    showStatus("This is shown at status");

}
}
```

- Save the file as **Sample. Java**
- 
- Compile the file using **javac Sample.java**

On successful compilation, execute the file using **appletviewer Sample.java**

The output appears as shown in following figure :



**Example 3** // Example for passing parameters

```
import java.awt.*; import java.applet.*;
```

```
/* <applet code="ParamDemo" width=300 height= 80> <param  
name=fontName value=Courier>
```

```
<param name=fontSize value=14> <param name=leading value = 2>
```

```
<param name=accountEnabled value= true> </applet>
```

```
*/
```

```
public class ParamDemo extends Applet
```

```
{
```

```
    String fontName; int fontSize; float leading; boolean active;
```

```
public void start()
```

```
{
```

```
    String param; fontName=getParameter("fontName"); if(fontName==null)
```

```
        fontName= "Not Found"; param=getParameter("fontSize"); try
```

```
    {
```

```
        if(param!=null)
```

```
            fontSize=Integer.parseInt(param); else
```

```
            fontSize=0;
```

```
    }
```

```
catch(NumberFormatException e)
```

```
{
```

```
    fontSize=-1;
```

```
}
```

```
param=getParameter("leading"); try
```

```
    {  
  
        if(param!=null) leading=Float.valueOf(param).floatValue(); else  
  
        leading=0;  
  
    }  
catch(NumberFormatException e)  
{  
    leading=0;  
}  
  
param=getParameter("accountEnabled"); if (param!=null)  
  
active =Boolean.valueOf(param).booleanValue();  
}
```

```
public void paint(Graphics g)
{

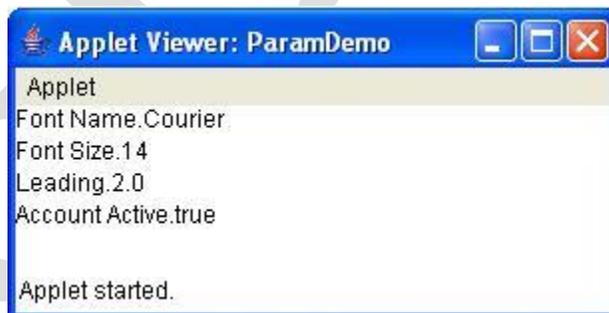
    g.drawString("Font Name." + fontName,0,10);
    g.drawString("Font Size." + fontSize,0,26);
    g.drawString("Leading." + leading,0,42);
    g.drawString("Account Active." + active,0,58);

}
}
```

- Save the file as **ParamDemo. Java**
- 
- Compile the file using **javac ParamDemo.java**

On successful compilation, execute the file using **appletviewer ParamDemo.java**

The output appers as shown in following figure :



**Example 4** // Example for `getDocumentBase()` & `getCodeBase()`

```
import java.awt.*;
import java.applet.*;
import java.net.*;

/* <applet code="Bases" width=300 height=
50> </applet>

*/

public class Bases extends Applet
{
    public void paint(Graphics g)
    {
        String msg;
        URL url= getCodeBase();

        msg= "Code Base:" +url.toString();
        g.drawString(msg,10,20);

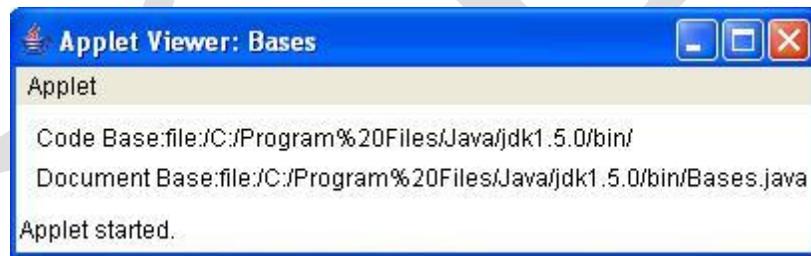
        url= getDocumentBase();
```

```
        msg= "Document Base:" +url.toString();  
        g.drawString(msg,10,40);  
    }  
}
```

- Save the file as **Bases. Java**
- 
- Compile the file using **javac Bases.java**

On successful compilation, execute the file using **appletviewer Bases.java**

The output appers as shown in following figure :



## **Graphical User Interface (GUI)**

### **Unit Structure**

- 11.1 GUI Components
- 11.2 Interface and Classes of AWT Package
  - 11.2.1 Labels
  - 11.2.2 Buttons
  - 11.2.3 Check Boxes
  - 11.2.4 Radio Button
  - 11.2.5 Text Area
  - 11.2.6 Text Field
  - 11.2.7 Scrollbar
  - 11.2.8 Panels
- 11.3 Layout managers
- 11.4 Methods of AWT

## Introduction

A type of user interface item that allows people to interact with programs in more ways than typing such as computers and many hand-held devices such as mobile phones is called a **graphical user interface (GUI)**. A *GUI* offers graphical icons, and visual indicators, as opposed to text-based interfaces. This helps to develop more efficient programs that are easy to work with. The user can interact with the application without any problem.

The GUI application is created in three steps. These are:

- Add components to Container objects to make your GUI.
- Then you need to setup event handlers for the user interaction with GUI.
- Explicitly display the GUI for application.

### 11.1 GUI Components

It is visual object and the user interacts with this object via a mouse or a keyboard. Components included, can be actually seen on the screen, such as, buttons, labels etc. Any operation that is common to all GUI components are found in class Component. Different components are available in the Java AWT (Abstract Window Toolkit) package for developing user interface for your program.

A class library is provided by the Java programming language which is known as Abstract Window Toolkit (AWT). The Abstract Window Toolkit (AWT) contains several graphical widgets which can be added and positioned to the display area with a layout manager.

AWT is a powerful concept in JAVA. AWT is basically used to develop for GUI application building. AWT is platform dependant. That means your **.class** file after the program compilation is platform independent but the look of your GUI application is platform dependant. AWT copies GUI component from local macines operating system. That means your applications look will differ in MAC operating system, as you have seen in WINDOWS operating system.

### **11.2 Interface and Classes of AWT Package:**

Some of the Classes Interfaces of AWT package are explained below

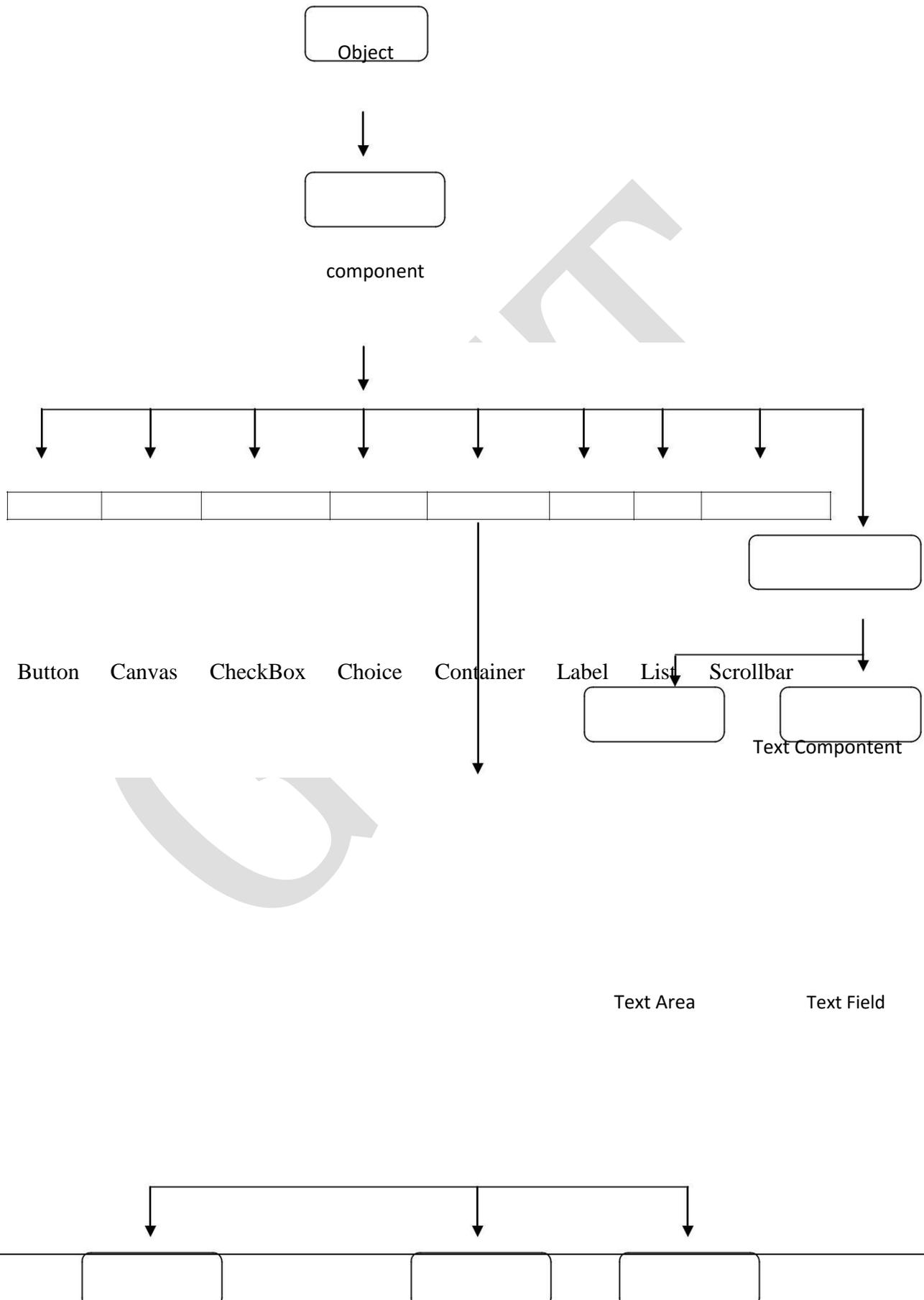
<b>Interfaces</b>	<b>Descriptions</b>
<b>ActionEvent</b>	This interface is used for handling events.
<b>Adjustable</b>	This interface takes numeric value to adjust within the bounded range.

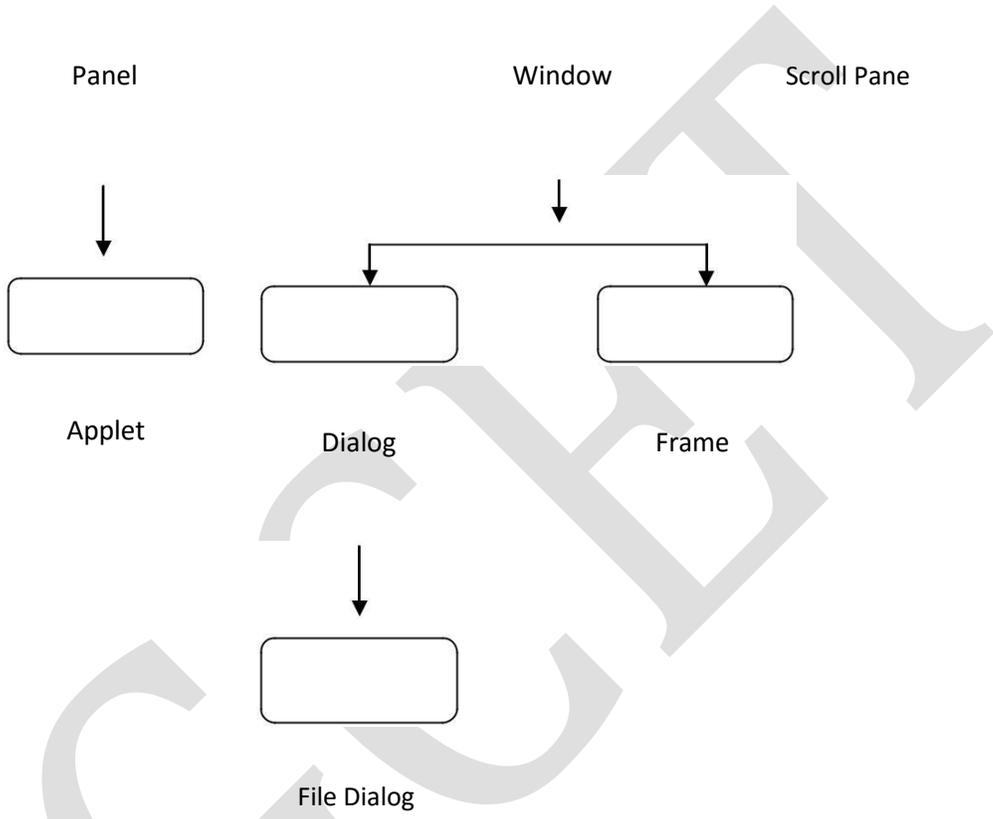
<b>Composite</b>	This interface defines methods to draw a graphical area. It combines a shape, text, or image etc.
<b>CompositeContext</b>	This interface allows the existence of several contexts simultaneously for a single composite object. It handles the state of the operations.
<b>ItemSelectable</b>	This interface is used for maintaining zero or more selection for items from the item list.
<b>KeyEventDispatcher</b>	The KeyEventDispatcher implements the current KeyboardFocusManager and it receives KeyEvents before dispatching their targets.
<b>KeyEventPostProcessor</b>	This interface also implements the current KeyboardFocusManager. The KeyboardFocusManager receives the KeyEvents after that dispatching their targets.
<b>LayoutManager</b>	It defines the interface class and it has layout containers.
<b>LayoutManager2</b>	This is the interface extends from the LayoutManager and is subinterface of that.
<b>MenuContainer</b>	This interface has all menu containers.
<b>Paint</b>	This interface is used to color pattern. It used for the Graphics2D operations.
<b>PaintContext</b>	This interface also used the color pattern. It provides an important color for the Graphics2D operation and uses the ColorModel.
<b>PaintGraphics</b>	This interface provides print a graphics context for a page.

<b>Shape</b>	This interface used for represent the geometric shapes.
<b>Stroke</b>	This interface allows the Graphics2D object and contains the shapes to outline or stylistic representation of outline.
<b>Transparency</b>	This interface defines the transparency mode for implementing classes.

GCE

Class hierarchy of AWT classes can be given as follows.





**Some of the AWT components are explained below.**

### **11.2.1 Labels:**

This is the simplest component of Java Abstract Window Toolkit. This component is generally used to show the text or string in your application and label never perform any type of action. Syntax for defining the label only and with justification:

```
Label label_name = new Label ("This is the label text.");
```

above code simply represents the text for the label.

```
Label label_name = new Label ("This is the label text. |", Label.CENTER);
```

Justification of label can be left, right or centered. Above declaration used the center justification of the label using the Label.CENTER.

### **Example for Label.**

```
import java.awt.*;
```

```
import java.applet.Applet;
```

```
/*<applet code="LabelTest" width=200  
height=100> </applet>
```

```
*/
```

```
public class LabelTest extends Applet
```

```
{
```

```
•  
•  
public void init()  
{  
  
    add(new Label("A  
label")); // right justify  
    next label  
  
    add(new Label("Another label", Label.RIGHT));  
}  
}
```

Save the file as **LabelTest.java**

Compile the file using **javac LabelTest.java**

On successful compilation, execute the file using **appletviewer LabelTest.java**

The output appears as shown in following figure :



### 11.2.2 Buttons:

This is the component of Java Abstract Window Toolkit and is used to trigger actions and other events required for your application. The syntax of defining the button is as follows:

```
Button button_name = new Button ("This is the label of the button.");
```

- 
- 

You can change the Button's label or get the label's text by using the Button.setLabel (String) and Button.getLabel () method. Buttons are added to its container using the, add (button\_name) method.

### Example for Buttons:-

```
import java.awt.*;
```

```
import java.applet.Applet;
```

```
/*<applet code="ButtonTest" width=200  
height=100> </applet>
```

```
*/
```

```
public class ButtonTest extends Applet
```

```
{
```

```
public void init()
```

```
{
```

```
    Button button = new Button
```

```
    ("OK"); add (button);
```

```
}
```

```
}
```

Save the file as **ButtonTest. Java**

Compile the file using **javac ButtonTest.java**

- On successful compilation, execute the file using **appletviewer ButtonTest.java**

The output appears as shown in following figure :



Note that in the above example there is no event handling added; pressing the button will not do anything.

### 11.2.3 Check Boxes:

This component of Java AWT allows you to create check boxes in your applications. The syntax of the definition of Checkbox is as follows:

```
Checkbox checkbox_name = new Checkbox ("Optional check box  
1", false);
```

Above code constructs the unchecked Checkbox by passing the boolean valued argument *false* with the Checkbox label through the Checkbox() constructor. Defined Checkbox is added to its container using add (checkbox\_name) method. You can change and get the checkbox's label using the setLabel (String) and getLabel () method. You can also set and get the state of the checkbox using the setState (boolean) and getState () method provided by the Checkbox class.

### Example for Check Boxes:-

```
import java.awt.*;
import java.applet.Applet;

/*<applet code="CheckboxTest" width=200
height=100> </applet>

*
public class CheckboxTest extends Applet
{
    public void init()
    {
        Checkbox m = new Checkbox ("Allow Mixed
        Case"); add (m);
    }
}
```

- Save the file as **CheckboxTest.java**
- 
- Compile the file using **javac CheckboxTest.java**

On successful compilation, execute the file using **appletviewer CheckboxTest.java**

The output appears as shown in following figure :



### 11.2.4 Radio Button:

Radio buttons are a bunch of option boxes in a group. Only one of them can be checked at a time. This is useful if you need to give the user a few options where only one will apply. This is the special case of the Checkbox component of Java AWT package. This is used as a group of checkboxes whose group name is same. Only one Checkbox from a Checkbox Group can be selected at a time.

Syntax for creating radio buttons is as follows:

```
CheckboxGroup chkboxgp = new CheckboxGroup ();  
add (new Checkbox ("chkboxname", chkboxgp,  
value);
```

—Value1 in the second statement can only be true or false. If you mention more than one true value for checkboxes then your program takes the last true and shows the last check box as checked.

#### **Example for Radio Buttons.**

```
import java.awt.*;  
import java.applet.Applet;  
  
/*<applet code="Rbutton" width=200  
height=100> </applet>  
*/  
  
public class Rbutton extends Applet  
{  
    public void init()  
    {
```

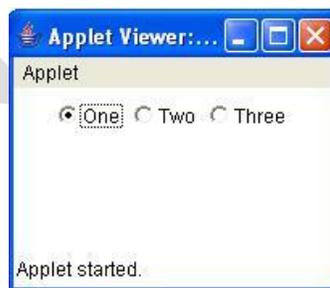
```
CheckboxGroup chkgp = new CheckboxGroup ();  
add (new Checkbox ("One", chkgp, false));  
  
add (new Checkbox ("Two", chkgp, false));  
add (new Checkbox ("Three",chkgp, false));  
  
}  
}
```

In the above code we are making three check boxes with the label "One", "Two" and "Three".

- Save the file as **Rbutton. Java**
- Compile the file using **javac Rbutton.java**

On successful compilation, execute the file using **appletviewer Rbutton.java**

The output appers as shown in following figure :



### 11.2.5 Text Area:

This is the text container component of Java AWT package. The Text Area contains plain text.

TextArea can be declared as follows:

```
TextArea txtArea_name = new TextArea ();
```

You can make the Text Area editable or not using the `setEditable (boolean)` method. If you pass the boolean valued argument *false* then the text area will be non-editable otherwise it will be editable. The text area is by default in editable mode. Texts are set in the text area using the `setText (string)` method of the `TextArea` class.

### Example for Text Area:-

```
import java.awt.*;
```

```
import java.applet.Applet;
```

```
/*<applet code="TAreaTest" width=200
```

```
height=100> </applet>
```

```
*/
```

```
public class TAreaTest extends Applet
```

```
{
```

```
    TextArea disp;
```

```
    public void
```

```
    init()
```

```
    {
```

```
        disp = new TextArea("Code goes here", 10,
```

```
        30); add (disp);
```

```
    }
```

```
}
```

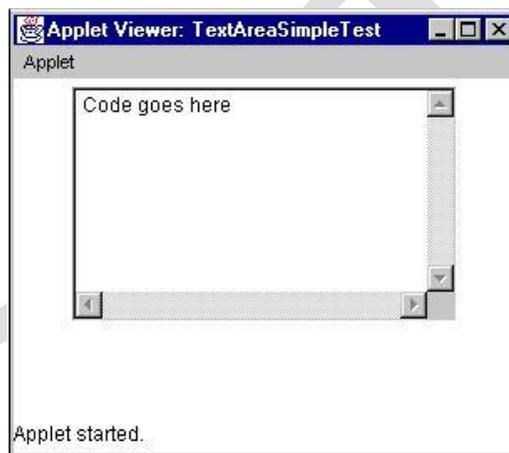
```
•  
•  
•
```

Save the file as **TAreaTest. Java**

Compile the file using **javac TAreaTest.java**

On successful compilation, execute the file using **appletviewer TAreaTest.java**

The output appears as shown in following figure :



### 11.2.6 Text Field:

This is also the text container component of Java AWT package. This component contains single line and limited text information. This is declared as follows:

```
TextField txtfield = new TextField (20);
```

You can fix the number of columns in the text field by specifying the number in the constructor.

In the above code we have fixed the number of columns to 20.

A displayed label object is known as the Label. Most of the times label is used to demonstrate the significance of the other parts of the GUI. It helps to display the functioning of the next text field. A label is also restricted to a single line of text as a button.

### Example for Text Field:-

```
import java.awt.*;
import java.applet.Applet;

/*<applet code="TFieldTest" width=200 height=100>
</applet>

*/

public class TFieldTest extends Applet
{
    public void init()
    {
        TextField f1 =

            new TextField("type something");
        add(f1);
    }
}
```

- Save the file as **TFieldTest.java**
- 
- Compile the file using **javac TFieldTest.java**

On successful compilation, execute the file using **appletviewer TFieldTest.java**

The output appears as shown in following figure :



### **11.2.7 Scrollbar:-**

Scrollbar is represented by a "slider" widget. The characteristics of it are specified by integer values which are being set at the time of scrollbar construction. Both the types of Sliders are available i.e. horizontal and vertical.

The example below shows the code for the scrollbar construction. The subtraction of scrollbar width from the maximum setting gives the maximum value of the Scrollbar. In the program code, '0' is the <<<<<< scrollbar.shtml initial value of the scrollbar, '8' is the width of the scrollbar.

#### **Example for Scrollbar**

```
import java.awt.*;

import java.applet.Applet;

/*<applet code="ScrollbarDemo" width=200 height=100>
</applet>

*/
public class ScrollbarDemo extends Applet

{
    public void init()
    {

        Scrollbar sb = new Scrollbar
            (Scrollbar.VERTICAL, 0, 8, -100, 100);

        add(sb);
    }
}
```



- Compile the file using **javac ScrollbarDemo.java**
- 
- On successful compilation, execute the file using **appletviewer ScrollbarDemo.java**

The output appears as shown in following figure :

### 11.2.8 Panels

A panel is an object which holds other objects. It's just a container to organize and arrange your GUI better. Once, you learn about Layout Managers you'll see why panels are a useful tool. For now, just know that they're useful. Here's an example of a set of buttons added into a panel:

```

Panel myPanel = new
Panel();
myPanel.add(helloButton);
myPanel.add(goodbyeButton
); add(myPanel);

```

It looks no different than if you just added the buttons regularly, but you'll see why you might want to use panels later on... This is what it looks like:



### 11.3 Layout managers

The layout manager are a set of classes that implement the **java.AWT.LayoutManager** interface and help to position the components in a container. The interface takes a task of laying out the child components in the container. The task is achieved by resizing and moving the child components. The advantages of this type of mechanism is that when the container is resized the layout manager automatically updates the interface

The basic layout managers includes:

- 1) **FlowLayout** : It is a simple layout manager that works like a word processor. It is also the default Layout manager for the panel. The flow layout lays out components linewise from left to right.

FlowLaout can be created using following constructors

- a. **FlowLaout()** : Constructs a new layout with centered alignment, leaving a vertical and horizontal gap.
- b. **FlowLayout(int align, int vgap, int hgap)** : Constructs a new flowlayout with the alignment specified, leaving a vertical and horizontal gap as specified.

Various methods can be used alog with the flow layout. For eg. `getAlignment()`, `getHgap()`, `getAlignment(int align)` etc.

### **Example for Flow Layout**

```
import java.awt.*;
import java.awt.event.*;
class FlowDemo extends Frame
{
    Button b1 = new Button("one");
    Button b2 = new Button("two");
    public FlowDemo(String s)
```

```
{
super(s);
setSize(400,400);

setLayout(new FlowLayout(FlowLayout.LEFT));
addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent e)
{
System.exit(0);
}
});
add(b1);
add(b2);

}
public static void main(String arg[])
{
Frame f=new
Frame(); f.show();

}
}
```

- 
- Save the file as **FlowDemo. Java**
- Compile the file using **javac FlowDemo.java**
- On successful compilation, execute the file using **java FlowDemo.java**

2) **Grid Layout** : It lays out components in a way very similar to spreadsheet(rows and columns). Specifying the number of rows and columns in grid creates the Grid layout.

Grid Layout can be created using following constructors

- a. **GridLayout()** : Creates a grid layout with a default of one column per component in a single row.
- b. **GridLayout(int rows, int cols, int hgap, int vgap)** : Creates a grid layout with the specified rows and columns and specified horizontal and vertical gaps.

Various methods can be used along with the Grid layout. For eg. `getColumns()`, `getRows()`, `getHgap()`, `getVgap()` etc.

### Example for Grid Layout

```
import java.applet.Applet; import
java.awt.*;

public class Grid1 extends Applet {
    LayoutManager Layout;

    Button [] Buttons;

    public Grid1 () { int i;

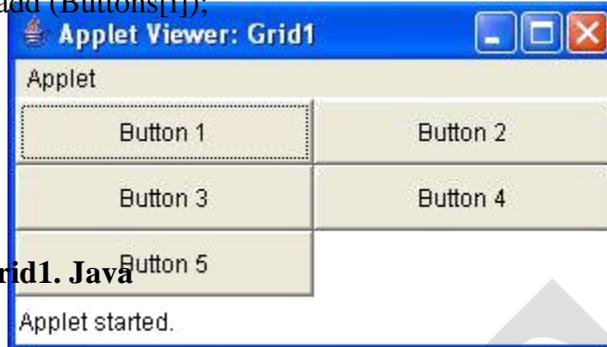
        Layout = new GridLayout (3, 2); setLayout
        (Layout);

        Buttons = new Button [5]; for (i = 0; i <
        5; ++i) {
```

```

Buttons[i] = new Button (); Buttons[i].setLabel
("Button " + (i + 1)); add (Buttons[i]);
}
}
}

```



- Save the file as **Grid1.java**
- 
- Compile the file using **javac Grid1.java**

On successful compilation, execute the file using **appletviewer Grid1.java**

The output appears as shown in following figure :

3) **BorderLayout** : It is the class that enables specification, i.e. where on the border of container each component should be placed. All areas need not be filled. The size of the areas will depend on the components they contain.

Border Layout can be created using following constructors

- BorderLayout()** : It creates a new border layout with no gap between the components.
- BorderLayout(int hgap, int vgap)** : It creates a border layout with the specified horizontal and vertical gap between components.

Various methods can be used along with the Border layout. For eg.  
 getHgap(), getVgap(), setHgap(int hgap), setVgap(int vgap)

### Example for BorderLayout

```

import java.awt.*; import
java.applet.*; import
java.util.*;

```

```
/*<applet code="BorderDemo" width=300 height=100> </applet>

*/

public class BorderDemo extends Applet

{
public void init()
{
setLayout(new BorderLayout());

add(new Button("This across the top"),BorderLayout.NORTH);

add(new Button("The Footer message might go here"),BorderLayout.SOUTH); add(new
Button("Right"),BorderLayout.EAST);

add(new Button("Left"),BorderLayout.WEST);

String msg=" This is border layout";

add(new TextArea(msg),BorderLayout.CENTER);
add(new Button("new"),BorderLayout.CENTER);

}
}
```

- Save the file as **BorderDemo. Java**
- 
- Compile the file using **javac BorderDemo.java**

On successful compilation, execute the file using **appletviewer BorderDemo.java**

The output appears as shown in following figure :



### 11.4 Methods of AWT:-

The common methods of AWT components are as follow:

**getLocation ()** - This method is used to get position of the component, as a Point. The usage of the method is shown below.

```
Point p = someComponent.getLocation (); int
```

```
x = p.x;
```

```
int y = p.y;
```

the x and y parts of the location can be easily accessed by using `getX ()` and `getY ()`. It is always efficient to use `getX ()` and `getY ()` methods.

For example,

```
int x = someComponent.getX();
```

```
int y = someComponent.getY();
```

**getLocationOnScreen ()** - This method is used to get the position of the upper-left corner of the screen of the component, as a Point. The usage of the method is shown below.

```
Point p = someComponent.getLocationOnScreen ();  
int x = p.x;
```

```
int y = p.y;
```

It is always advisable to use getLocation () method (if working on Java 2 platform).

**getBounds ()** - This method is used to get the current bounding Rectangle of component. The usage of the method is shown below.

```
Rectangle r = someComponent.getBounds ();  
int height = r.height;
```

```
int width = r.width;
```

```
int x = r.x;
```

```
int y = r.y;
```

if you need a Rectangle object then the efficient way is to use getX (), getY(), getWidth(), and getHeight() methods.

**getSize ()** - This method is used to get the current size of component, as a Dimension. The usage of the method is shown below.

```
Dimension d = someComponent.getSize ();
```

```
int height = d.height;
```

```
int width = d.width;
```

use `getWidth ()` and `getHeight ()` methods to directly access the width and height. You can also use `getSize ()` if you require a `Dimension` object.

```
For Example, int height = someComponent.getHeight();  
int width = someComponent.getWidth();
```

**setBackground(Color)/setForeground(Color)** - This method is used to change the background/foreground colors of the component

**setFont (Font)** - This method is used to change the font of text within a component.

**setVisible (boolean)** - This method is used for the visibility state of the component. The component appears on the screen if `setVisible ()` is set to true and if it's set to false then the component will not appear on the screen. Furthermore, if we mark the component as not visible then the component will disappear while reserving its space in the GUI.

**setEnabled (boolean)** - This method is used to toggle the state of the component. The component will appear if set to true and it will also react to the user. ON the contrary, if set to false then the component will not appear hence no user interaction will be there.

As discussed earlier a container is a component that can be nested. The most widely used Panel is the `Class Panel` which can be extended further to partition GUIs. There is a Panel which is used for running the programs. This Panel is known as `Class Applet` which is used for running the programs within the Browser.

### **Common Container Methods:-**

All the subclasses of the `Container` class inherit the behavior of more than 50 common methods of `Container`. These subclasses of the container mostly override the method of component. Some of the methods of container which are most widely used are as follow:

```
getComponents (); add();  
getComponentCount();  
getComponent(int);
```

### **ScrollPane:-**

The ScrollPane container provides an automatic scrolling of any larger component introduced with the 1.1 release of the Java Runtime Environment (JRE). Any image which is bigger in size for the display area or a bunch of spreadsheet cells is considered as a large object. Moreover there is no LayoutManager for a ScrollPane because only a single object exists within it. However, the mechanism of Event Handling is being managed for scrolling.

The example below shows the Scrollpane. This scrollpane demonstrates the scrolling of the large image. In the program code below, first of all we have created a scrollpane by creating its object, and then we have passed the parameter of image in it. We have also set the border layout as centre, as shown.

#### **Example for Scroll Pane**

```
import java.awt.*;  
import java.applet.*;  
  
/*<applet code="ScrollingImageDemo" width=200 height=100>  
</applet>  
  
*/  
  
class Scrollpane extends Component {  
    private Image image;  
    public Scrollpane(Image m)  
  
    {  
        image = m;  
  
    }  
}
```

```
public void paint(Graphics g)

{

    if (image != null)
        g.drawImage(image, 0, 0, this);

}

}

public class ScrollingImageDemo extends Applet

{

    public void init()

    {

        setLayout(new BorderLayout());

        ScrollPane SC = new ScrollPane(ScrollPane.SCROLLBARS_ALWAYS)

;

        Image mg = getImage(getCodeBase(), "cute-puppy.gif");

        SC.add(new Scrollpane(mg));

        add(SC, BorderLayout.CENTER);

    }

}


```

Save the file as **ScrollingImageDemo.java**

- Compile the file using **javac ScrollingImageDemo.java**
- On successful compilation, execute the file using **appletviewer ScrollingImageDemo.java**

## EventHandling

### 12.0 Introduction

Writing an applet that responds to user input, introduces us to event handling. We can make our applet respond to user input by overriding event handler methods in our applet. There are a variety of event handler methods which we will see further.

Each event must return a Boolean value (true or false), indicating whether the event should be made available to other event handlers. If you've processed an event (for example, keyDown) you might record the value, and return true to show that no other handlers should receive the event. If, however, your custom edit box can't process the character, it may want to return false to signal that other components (the panel or applet in which the component is hosted) should process it.

### 12.1 Event:

An **Event** is an object that describes a state change in a source. It can be generated as a consequence of a person interacting with the elements in a GUI. Some of the activities that cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse.

Events may also occur that are not directly caused by interactions with user interface. For e.g. an event may be generated when a timer expires, a counter exceeds a value, a software or hardware failure occurs, or an operation is completed.

### 12.2 Event Source:

An **event source** is the object that generates an event. This occurs when the internal state of that object changes in some way. Sources may generate more than one type of event. A source may register listeners in order for the listeners to receive notifications about a specific type of event. Each type of event has its own registration method. Example if you click a button an `ActionEvent` Object is generated. The object of the `ActionEvent` class contains information about the event.

In addition to GUI elements, other components such as an Applet, can generate Events. For e.g. you receive key and mouse events from an Applet.

Following is the table to describe some of the Event Sources.

<b>Event Sources</b>	<b>Description</b>
Button	Generates action events when the button is pressed.
Checkbox	Generates item events when the check box is selected or deselected.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Choice	Generates item events when the choice is changed.
MenuItem	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scrollbar	Generates adjustment events when the scrollbar is

	manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed , deactivated, deiconified, iconified, opened, or quit.

### 12.3 Event Classes

The „**EventObject**“ class is at the top of the event class hierarchy. It belongs to the java.util package. While most of the other event classes are present in java.awt.event package. The **getSource()** method of the EventObject class returns the object that initiated the event. The **getId()** method returns the nature of the event. For example, if a mouse event occurs, you can find out whether the event was click, a press, a move or release from the event object.

Following is the table to describe the Event Classes.

Event Class	Discription
ActionEvent	A semantic event which indicates that a component-defined action occurred.
AdjustmentEvent	The adjustment event emitted by Adjustable objects.
ComponentEvent	A low-level event which indicates that a component moved, changed size, or changed visibility (also, the root class for the other component-level events).
	A low-level event which indicates that a container's

ContainerEvent	contents changed because a component was added or removed.
InputEvent	The root event class for all component-level input events.
ItemEvent	A semantic event which indicates that an item was selected or deselected.
KeyEvent	An event which indicates that a keystroke occurred in a component.
MouseEvent	An event which indicates that a mouse action occurred in a component.
MouseEvent	An event which indicates that the mouse wheel was rotated in a component.
PaintEvent	The component-level paint event.
TextEvent	A semantic event which indicates that an object's text changed.
WindowEvent	A low-level event that indicates that a window has changed its status.

#### 12.4 Event Listener:

These are objects that define methods to handle certain type of events. An event source (for example a PushButton) can generate one or more type of events, and maintain a list of event listeners for each type of event. An event source can register listeners by calling addXListener type of methods. For example a Button may register an object for handling ActionEvent by calling addActionListener. This object would then need to implement the listener interface corresponding to ActionEvent, which is ActionListener.

So to set up the processing of events the following tasks must be done.

- I. For the GUI component (like pushbutton) associate a listener object class with the component by calling a method of type addXListener (See table below for list of methods).
- II. Define this listener object. It must implement the corresponding interface. The name of interface is of type EventListener. Table below gives list of event listeners.
- III. The object must define all the methods defined in the interface it is implementing. See table for the list of Event Listener methods defined in each Event Listener interface

<b>Interface</b>	<b>Description</b>
ActionListener	Define one method to receive action events
AdjustmentListener	Defines one method to receive adjustment events.
ComponentListener	Defines four methods to recognize when a component is hidden, moved, resized, or shown.
ContainerListener	Defines two methods to recognize when a component is added to or removed from a container.
Focus Listener	Defines two methods to recognize when a component gains or loses keyboard focus.
ItemListener	Defines one method to recognize when the state of an item is changes.
KeyListener	Defines 3 methods to recognize when a key is pressed, released, or typed.
MouseListener	Defines 5 methods to recognize when the mouse is

	clicked, enters a component, exits a component, is pressed, or is released
MouseMotionListener	Defines two methods to recognize when the mouse is dragged or moved.
MouseWheelListener	Defines one method to recognize when mouse wheel is moved.
TextListener	Defines one method to recognize when a text value changes.
WindowFocusListener	Defines two methods to recognize when window gains or loses focus.
WindowListner	Defines 7 methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

## 12.5 EXAMPLES

### 1) Example for MouseEvents & MouseListener

```
import java.awt.*; import
java.awt.event.*; import
java.applet.*;
```

```
/*<applet code= "mouseEvent" width=400 height=300>
```

```
</applet?
```

```
*/
```

```
public class mouseEvent extends Applet implements MouseListener, MouseMotionListener
```

```
{
```

```
    public void init ()
```

```
    {
```

```
        addMouseListener (this);
        addMouseMotionListener (this);
    }
    public void mouseClicked(MouseEvent e)
    {
        showStatus ("Mouse has been clicked at " + e.getX()+ "," + e.getY());
    }

    public void mouseEntered (MouseEvent e)

    {
        showStatus ("Mouse has been Entered at " + e.getX()+ "," + e.getY());

        // For loop: to make sure mouse entered is on status bar for a few sec for (int
        i= 0; i<1000000; i++);

    }
    public void mouseExited (MouseEvent e)
    {
        showStatus ("Mouse has been Exited at " + e.getX()+ "," + e.getY());
    }

    public void mousePressed (MouseEvent e)
    {
        showStatus ("Mouse pressed at " + e.getX()+ "," + e.getY());
    }

    public void mouseReleased (MouseEvent e)
    {
        showStatus ("Mouse released at " + e.getX()+ "," + e.getY());
    }
}
```

```
public void mouseDragged (MouseEvent e)
{
    showStatus ("Mouse dragged at " + e.getX()+ "," + e.getY());
}

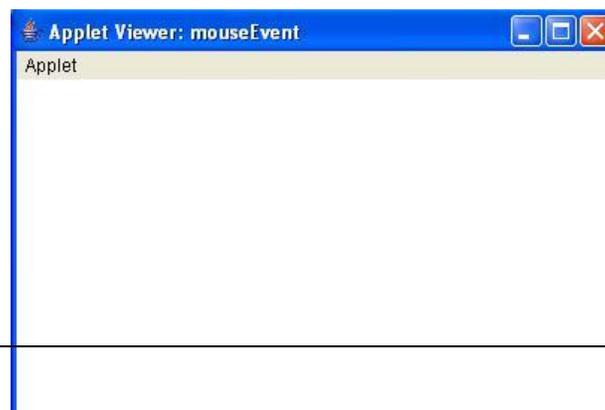
public void mouseMoved(MouseEvent e)
{
    showStatus ("Mouse moved at " + e.getX()+ "," + e.getY());
}
//public void paint(Graphics g)
//    {
//g.drawString(msg, e.getX(), e.getY());
//
//
}
```

- 
- 
- Save the file as **mouseEvent. Java**

Compile the file using **javac mouseEvent.java**

On successful compilation, execute the file using **appletviewer mouseEvent.java**

The output appers as shown in following figure :



## 2) Example for Key events and KeyListener.

```
import java.awt.*; import
java.awt.event.*; import
java.applet.*;

/*<applet code="keyTest" width =400 height=300> </applet>

*/
public class keyTest extends Applet implements KeyListener
{

    public void init()
    {

        Label lab = new Label ("Enter Characters :"); add (lab);

        TextField tf = new TextField (20); add (tf);

        tf.addKeyListener(this);

    }
    public void keyPressed(KeyEvent e)
    {
        showStatus("key Down");
    }
}
```

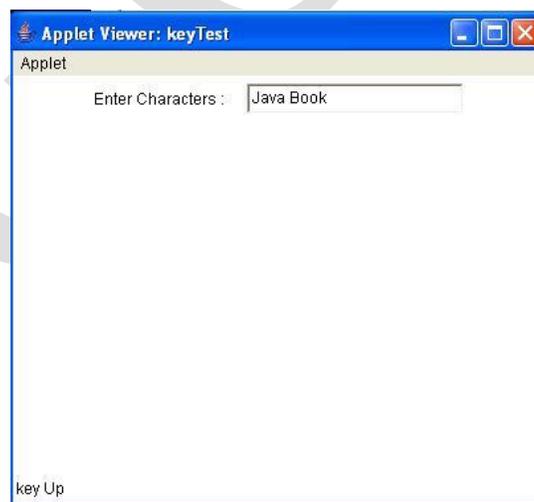
```
}  
public void keyReleased(KeyEvent e)  
{  
  
    showStatus("key Up");  
  
}  
public void keyTyped(KeyEvent e)  
{  
    showStatus(" Recently typed characters are : " + e.getKeyChar());  
}  
}
```

- 
- Save the file as **keyTest. Java**
- 

Compile the file using **javac keyTest.java**

On successful compilation, execute the file using **appletviewer keyTest.java**

The output appers as shown in following figure :



Example for Button Event and Action Listener

```
import java.awt.*; import
java.awt.event.*;
import java.applet.Applet;

/*

<applet code = "ButtonEvent" height = 400 width = 400>
</applet>

*/

public class ButtonEvent extends Applet implements ActionListener
{
    Button b;

    public void init()
    {

        b = new Button("Click me");
        b.addActionListener(this); add (b);

    }

    public void actionPerformed (ActionEvent e)
    {
        //    If the target of the event was our //Button
        //    In this example, the check is not
        //    Truly necessary as we only listen//to
        //    A single button

        if(e.getSource () == b)
        {
```

```

getGraphics().drawString("OUCH Buddy",20,20);
}
}
}

```



- Save the file as **ButtonEvent.java**
- 
- 
- Compile the file using **javac ButtonEvent.java**

On successful compilation, execute the file using **appletviewer ButtonEvent.java**

The output appears as shown in following figure :

**12.6 Handling Windows Events:**

When you use interfaces for creating listeners, the listener class has to override all the methods that are declared in the interface. Some of the interfaces have only one method, whereas others(windowListener) have many. So even if you want to handle only a single event, you have to override all the methods. To overcome this, the event packages provide seven adapter classes, which we will see shortly. Now coming back to handle window-related events, you need to register the listener object that implements the windowListener interface. The WindowListener interface contains a set of methods that are used to handle window events.

Category	Event	Method
Windows Events	The user clicks on the cross button.	void windowClosing (WindowEvent e)
	The window opened for the first time.	void windowOpened (WindowEvent e)

	The window is activated.	void windowActivated (WindowEvent e)
	The window is deactivated.	void windowDeactivated (WindowEvent e)
	The window is closed.	void windowClosed (WindowEvent e)
	The window is minimized	void windowIconified (WindowEvent e)
	The window maximized	void windowDeiconified (WindowEvent e)

#### 4) example for Window Events

```
import java.awt.*; import  
java.awt.event.*;
```

```
Class OurWindowListener implements WindowListener
```

```
{
```

```
    //Event handler for the window closing event public void  
    windowClosing (WindowEvent we)
```

```
    {  
        System.exit(0);  
    }  
    public void windowClosed (windowEvent we)  
    {  
    }  
    public void windowOpened (windowEvent we)  
    {  
    }  
    public void windowActivated (windowEvent we)  
    {  
    }  
    public void windowDeactivated (windowEvent we)  
    {  
    }  
    public void windowIconified (windowEvent we)  
    {  
    }  
    public void windowDeiconified (windowEvent we)  
    {  
    }  
}
```

```
public class MyFrame extends Frame  
{  
    Button b1;  
    // Main Method  
    public static void main (String arg[])  
    {  
        MyFrame f = new MyFrame();  
    }  
}
```

```
}
```

```
//Constructor for the event derived class
```

```
public MyFrame()
```

```
{
```

```
    Super ("—Windows Events-Title");
```

```
    b1 = new button("—Click Me");
```

```
    //place the button object on the window
```

```
    add("—center",b1);
```

```
    //Register the listener for the button ButtonListener
```

```
    listen = new ButtonListener();
```

```
    b1.addActionListener(listen);
```

```
    //Register a listener for the window.
```

```
    OurWindowListener wlisten = new OurWindowListener();
```

```
    addWindowListener(wlisten);
```

```
    //display the window in a specific size
```

```
    setVisible(true);
```

```
    setSize(200,200);
```

```
    }//end of frame class
```

```
//The Listener Class
```

```
Class ButtonListener implements ActionListener
```

```
{
```

```

//Definition for ActionPerformed() method public
void ActionPerformed(ActionEvent evt)

{
    Button source = (Button)evt.getSource();
    Source.setLabel(—Button Clicked, Buddy!!);
}
}
}

```

In the above example MyFrame class makes a call to the addWindowListener() method, which registers object for the window. This enables the application to handle all the window-related events. When the user interacts with the application by clicking close button, maximizing or minimizing a WindowEvent object is created and delegated to the pre-registered listener of the window. Subsequently the designated event-handler is called.

In the above example, the class OurWindowListener has methods that do not contain any code. This is because the windowListener interface contains declarations for all these methods forcing you to override them.

### **12.7 Adapter Classes:**

Java provides us with adapter classes that implement the corresponding listener interfaces containing one or more methods. The methods in these classes are empty. The Listener class that you define can extend the Adapter class and override the methods that you need. The adapter class used for WindowListener interface is the WindowAdapter class.

So you can simplify the above code (example 2) using Adapter class in the following manner:

Example :Save as MyFrames.java and compile.

```
import java.awt.*; import
java.awt.event.*;
```

Class MyFrames extends frame

```
{
    public static void main(String arg[])
    {
        MyFrames f = new MyFrames();
    }

    //constructor of the Frame derived class
    public MyFrames
    {

        //Register the Listener for the window
        super(—The Window Adapter Sample);

        MyWindowListener mlisten = new MyWindowListener();
        addWindowListener(mlisten);

        setVisible(true);
    }
}
```

Class MyWindowListener extends WindowAdapter

```
{

    //event handler for windows closing event public
    void windowClosing(WindowEvent we)

    {
```

```

MyFrames f;

f = (MyFrames)we.getSource();
f.dispose();
System.exit(0);
}
}

```

The Following is a list of Adapter classes and Listener Interfaces In Java:

Event Category	Interface Name	Adapter Name	Method
Window	Window Listener	Window Adapter	Void windowClosing (WindowEvent e)
			Void windowOpened (WindowEvent e)
			Void windowActivated (WindowEvent e)
			Void windowDeactivated (WindowEvent e)
			Void windowClosed (WindowEvent e)
			Void

			windowIconified (WindowEvent e)
			Void windowDeiconified (WindowEvent e)
Action	ActionListener		Void actionPerformed(ActionEvent e)
Item	ItemListener		Void itemStateChanged(ItemEvent e)
Mouse Motion	MouseMotionListener	MouseMotionAdapter	Void mouseDragged(MouseEvent e)
			Void mouseMoved(MouseEvent e)
Mouse Button	MouseListener	MouseAdapter	Void mousePressed(MouseEvent e)
			Void mouseReleased(MouseEvent e)

			Void mouseEntered(MouseEvent e)
			Void mouseClicked(MouseEvent e)
			Void mouseExited(MouseEvent e)
Key	KeyListener	KeyAdapter	Void keyPressed(KeyEvent e)
			Void keyReleased(KeyEvent e)
			Void keyTyped(KeyEvent e)
Focus	FocusListener		Void focusGained(FocusEvent e)
			Void focusLost(FocusEvent e)

component	ComponentListener	ComponentAdapter	Void componentMoved(ComponentEvent e)
			Void componentResized(ComponentEvent e)
			Void componentHidden(ComponentEvent e)
			Void componentShown(ComponentEvent e)

{mospagebreak title=Dissecting Java As Far As Inner Classes} Inner Classes:

Inner classes are classes that are declared within other classes. They are also known as nested classes and provide additional clarity to the program. The scope of the inner class is limited to the class that encloses it. The object of the inner class can access the members of the outer class. While the outer class can access the members of the inner class through an object of the inner class.

Syntax:

```
class
{
```

```
class
{

}
//other attributes and methods
}
```

Example: Save as MyFrame.java then compile and excute the program.

```
import java.awt.*; import
java.awt.event.*;
Class MyFrame extends Frame
{
//inner class declaration
class MyWindowListener extends MyAdapter
{

//event handler for windows closing event public
void windowClosing(WindowEvent w)
{
MyFrame frm;

frm = (MyFrames)w.getSource();
frm.dispose();
System.exit(0);
}

public static void main(String arg[])

{
MyFrame frm = new MyFrame();
}
```

```

//constructor of the Frame class
public MyFrames

{

    //Register the Listener for the window
    super(—Illustration For Inner or Nested Classes!);
    //creating an object of inner class

    MyWindowListener wlisten = new MyWindowListener();
    addWindowListener(wlisten);

    setVisible(true);
    setSize(100,100);
}
}

```

The above example code declares an object of the inner class in the constructor of an outer class. To create an object of the inner class from an unrelated class, you can use the new operator as if it were a member of the outer class.

Example:

```

MyFrame frame = new MyFrame(—Title!);
Frame.MyWindowsListener listen = new MyFrame().MyWindowListener();

```

You can create a class inside a method. The methods of the inner class can have access to the variables define in the method containing them. Inner class must be declared after the declaration of the variables of the method so those variables are accessible to the inner class.

Example : Save As RadioTest.java, Compile And View Using Appletviewer

In this Applet example we examine MouseAdapters, and its methods like mouseClicked(). Plus ItemListener interface implementation and itemStateChanged() method and use getItem() method to display the item the user as selected in the Applet's status bar using the showStatus()method. We will use interface components like checkbox, which are of two types-exclusive checkboxes (which means only one among the group can be selected) also called Radio Buttons. We also use non-inclusive checkboxes, which can be selected independently. The Choice class implements the pop-up menu that allows users to select items from a menu. This UI component displays the currently selected item with a arrow to its right.

```
/*
```

```
<applet code = "RadioTest.class" height = 300 width = 300 >
```

```
</applet>
```

```
*/
```

```
import java.awt.*; import  
java.awt.event.*; import  
java.applet.*;
```

```
public class RadioTest extends Applet
```

```
{
```

```
    public void init()
```

```
    {
```

```
        CheckboxGroup cbg = new CheckboxGroup();
```

```
        // Checkbox(label, specific checkgroup, checked:boolean)
```

```
        Checkbox c1 = new Checkbox("Black and White",cbg,false);
```

```
        Checkbox c2 = new Checkbox("Color",cbg,false);
```

```
//adding mousetlistener to the corresponding //
    component to trap the event

c1.addMouseListener(new check1());
c2.addMouseListener(new check2());

//adding components to the container
add(c1);

add(c2);

//To create a Choice Menu(say to list the various choices)
//    a Choice Object is instantiated.

// In short-Choice() constructor creates a new choice menu
//& you add items using addItem()

    Choice c = new Choice();
    c.add("LG");
    c.add("Onida");
    c.add("BPL");
    c.add("Samsung");
    c.add("Philips");
    c.add("Sony");

// adding ItemListener to choice then adding it to the container
    c.addItemListener(new Ch());

    add(c);
}
```

Class check1 extends MouseAdapter

```
{  
    Public void mouseClicked(MouseEvent e)  
    {  
        showStatus("You have selected Black & White TV option");  
    }  
}
```

Class check2 extends MouseAdapter

```
{  
    Public void mouseClicked(MouseEvent e)  
    {  
        showStatus("You have selected Color TV option");  
    }  
}
```

Class Ch implements ItemListener

```
{  
Public void itemStateChanged(ItemEvent e)  
{  
    String s =(String)e.getItem();  
    showStatus("You have selected" + s + " brand for your TV");  
}  
}  
}
```

# Swing

## 13.1 Introduction to JFC (Java Foundation Classes)

The earlier versions of java were released with some simple libraries. JDK1.2 was introduced with a new set of packages – the java foundation classes, or JFC – that includes an improved user interface called the **swing** components.

The JFC were developed, to address the shortcomings of AWT(Abstract Windowing Toolkit). The development of JFC was unique. JFC 1.2 is an extension of the AWT, not a replacement for it. The JFC visual components extend the AWT container class. The methods contained in the component and container classes that AWT programmers are familiar with are still valid for JFC visual classes.

The AWT user interface classes are now superseded by classes provided by the JFC. The support classes play an important role in JFC applications . AWT support classes , those that do not create a native window are not replaced by JFC classes.

## 13.2 Swing

Swing components facilitate efficient graphical user interface (GUI) development. These components are a collection of lightweight visual components. Swing components contain a replacement for the heavyweight AWT components as well as complex user-interface components such as trees and tables.

Swing components contain a pluggable look and feel(PL&F). This allows all applications to run with the native look and feel on different platforms. PL&F allows applications to have the same behavior on various platforms. JFC contains operating systems neutral look and feel. Swing

components do not contain peers. Swing components allow mixing AWT heavyweight and swing lightweight components in an application. The major difference between lightweight and heavyweight components is that lightweight components can have transparent pixels while heavyweight components are always opaque. Lightweight components can be non-regular while heavyweight components are always rectangular.

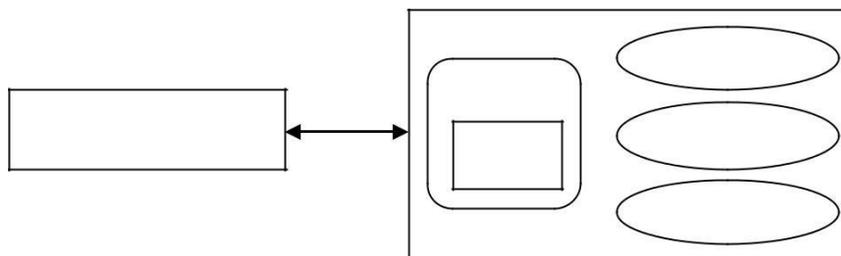
Swing components are JavaBean compliant. This allows components to be used easily in a Bean aware applications building program. The root of the majority of the swing hierarchy is the Jcomponent class. The class is an extension of the AWT container class .

### Difference between Swing and AWT

Swing	AWT
Swing component does not need any native code to implement.	AWT component can be implementing with code.
Swing lets you specify which look and feel your programs GUI uses.	AWT components always have the look and feel of the native platform.
Swing components don't have to be rectangular. For ex. Buttons can be rounded.	AWT components are always rectangular.

The swing architecture is shown in the figure given below.

Application code



JFC

Java 2D

Swing

Drag & drop

AWT

Accessibility

GCEET

Swing components comprises of a large percentage of the JFC release. The swing component toolkit consists of over 250 pure java classes and 75 interfaces contained in about 10 packages. They are used to build lightweight user interface. Swing consists of user interface(UI) classes and non user interface classes. The non-UI classes provide services and other operations for the UI classes.

### **Swing packages:**

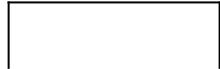
Some of the Swing packages are given below.

- Javax.swing.plaf.basic : Contains classes that define the default look and feel of swing components.
- Javax.swing.border : Contains border interface and their related interfaces.
- Javax.swing.event: Define events specific to swing components.
- Javax.swing.plaf.multi: Consist of multiplexing UI classes
- Javax.swing.plaf: Consist of classes that provide swing components with pluggable look and feel capabilities.
- Javax.swing.table: Contains classes and interfaces specific to the swing table components
- Javax.swing.text: Contains classes and interfaces for text manipulation components contained in swing toolkit.
- Javax.swing.tree: Contains classes that are used with the swing tree component
- Javax.swing.undo: contains interfaces and classes required to implement the undo functionality.

### 13.3 Swing Features :

- **MVC Architecture:** The user can provide his own data-model for a component by subclassing the Model class or by implementing the appropriate interface. The Model-View-Controller (MVC) architecture is used consistently throughout the swing component set. The view and controller parts of the architecture are combined in the component.
- **Nested Containers:** Swing was designed to manage nested containers gracefully. The main heavyweight containers(JWindow, JFrame etc.) as well as the major `_lightweight` containers(JInternalFrame and JComponent) all delegate their operations to a JRootPane. This commonly produces high degree of regularity in container nesting. In particular since the fundamental component class(JComponent) contains a JRootPane, virtually any component can be nested within another.
- **Keystroke Handling:** A user can register interest in a particular combination of keystrokes by creating a keystroke object and registering it with the component. When the keystroke combination is registered along with its association action, certain conditions have to be specified. These determine the time of initiation of the action.
- **Action Objects:** action interface objects provide a single point of control for program actions. An example of this would be a toolbar icon and a menu item referencing the same Action objects. When action object disabled, the GUI items that reference it are automatically disabled.
- **Virtual Desktops:** The JdesktopPane and JInternalFrame classes can be used to create a virtual desktop or multiple document interface. A JInternalFrame can be specified as
- cognizable, expandable or closable, while the JDesktopPane Provides real estate for them to operate in.
-

**Pluggable look and feel:** The user can select a look and feel and this can be plugged in. An interface made of Swing components can look like a Win32 app, a Motif app. It can use the new Metal look and feel.



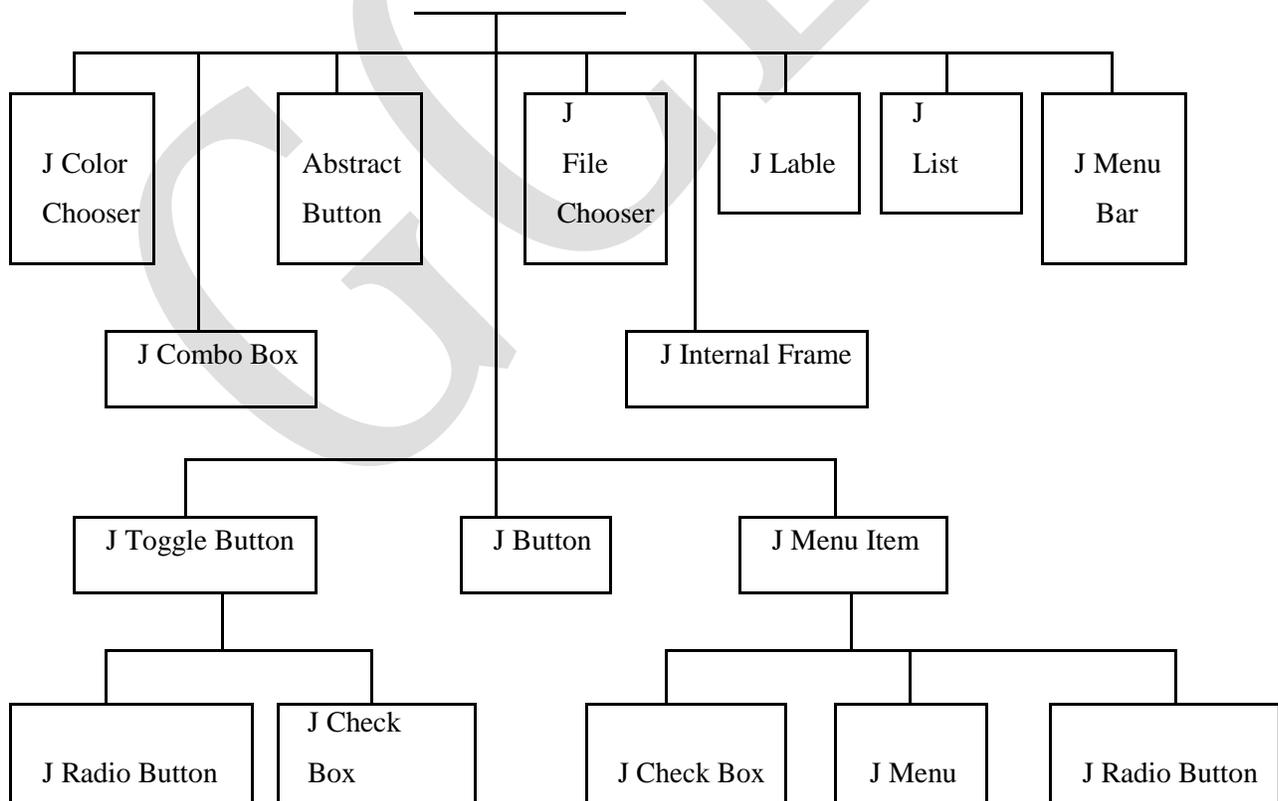
**Wide variety of components:** Class names that starts with J are the components that are added to and application. For ex. JButton, JList, JPanel.

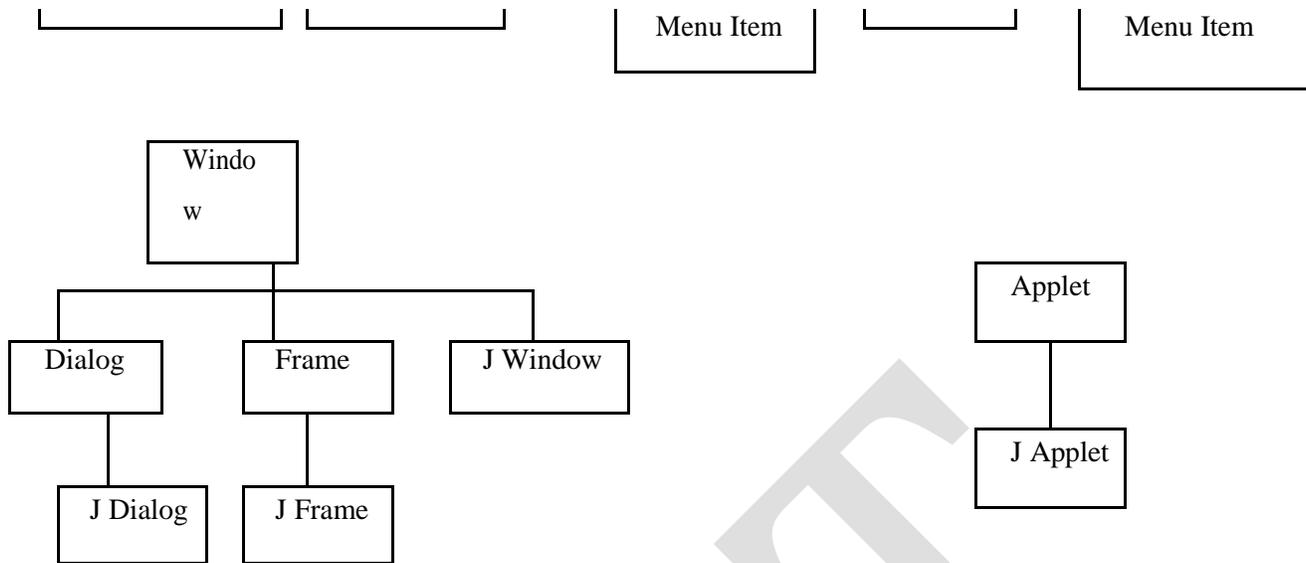
### 13.4 JComponent

The JComponent class is the root of the visual component class hierarchy in the JFC. The visual components are known as the —J classes. The functionality contained in the JComponent class is available to all the visual components contained in the JFC. The JComponent class is repository of functionality for all visual components.

The JComponent class is at the top of the hierarchy of all visual components contained in the JFC. The hierarchy is shown in the following figure.

JComponent





### 13.5 JApplet

The JApplet class is an extended version of the AWT applet class that adds support for root panes and other panes.. This class is the preferred entry point when creating applets that contain JFC components. The components are added to the ContentPane.

**The constructor that can be used to create a JApplet are listed below:**

- JApplet() : It creates a swing applet instance

**Some of the methods that can be used in conjunction with the JApplet is given below:**

- createRootPane(): Called by the constructor methods to create the default root pane.
- getContentPane() : Returns the content pane object for the applet
- getGlassPane() : Returns the glass pane object for the applet

- 
- getJMenuBar( ) : Returns the menu bar set on the
- applet setContentPane( ) : sets the content pane properly
- setGlassPane( ) : sets the glass pane properly
- 
- setLayout(LayoutManagermanager) : By default the layout of this component may not be set, the layout of its contentPane should be set instead.

### 13.6 JFrame

**Frame windows:** A frame is a top-level window that contains a title, border, minimize and maximize buttons. JFC provides the JFrame class. This is used as a top-level-frame.

**JFrame :** A Swing frame is represented by the class JFrame, is an extension of the AWT Frame classes. It is the part of javax.swing package. A Swing frame is a container that functions as the main window for programs that use Swing components. An instance of the JFrame Class is a heavyweight component.

**The JFrame can be created using the constructors mentioned below:**

- JFrame() : Constructs a new frame that is initially invisible.
- JFrame(String title) : Constructs a new frame, initially invisible with the specified title.

**Some of the methods that may be used in conjunction with the JFrame() are listed below:**

- createRootPane() : Called by the constructor methods to create the default root
- pane frameInit() : Called by the constructor to init the JFrame properly.
- getContentPane() : Returns the content pane object for this frame
- 
- 
- 
- 
- 
-

getGlassPane() : Returns the glass pane object for this frame  
getJMenuBar() : Returns the menu bar set on this frame  
getLayeredPane() : Returns the layered pane for this frame  
setContentPane() : Sets the content pane property  
setGlassPane() : Sets the glass pane property  
setJMenuBar() : Sets the menu bar for the frame

- setLayout(LayoutManager manager) : By default the layout of this component may not be set, the layout of its contentPane should be set instead.

### **13.7 JPanel**

JPanel is a Swing lightweight container that is often used for grouping components within one of an applet or a frame. It can also be used to group other panels. The primary purpose of the class is to provide a concrete container for the JFC. The JPanel class is provided to give a concrete container class. Being an extension of the JComponent class, JPanel is a container and inherits the features contained in that class.

**The various constructors that can be used to create a JPanel are as given below.**

- JPanel() : Create a new JPanel with a double buffer and a flow layout.
- JPanel(Boolean is DoubleBuffered) : Create a new JPanel with FlowLayout and the specified buffering strategy.
- JPanel (LayoutManager layout) : create a buffered JPanel with the specified layout manager.
- JPanel(LayoutManager layout, boolean is DoubleBuffered) : Creates a new JPanel with the specified layout manager and buffering strategy.

**The methods supported by this class includes :**

- `getAccessibleContext()` : Gets the `AccessibleContext` associated with this `JComponent`.
- `getUIClassID()` : Returns a string that specifies the name of the L&F class that renders this component.
- `paramString()` : Returns a string representation of the corresponding `JPanel`.
- `update()` : Notification from the `UIFactory` that the L&F has changed.

### **13.8 JButtons, checkboxes and Radiobuttons**

**JButton:** `JButton` behaves in a way that is similar to `Button`. It can be added to `JPanel` and its actions can be monitored via the `ActionListener`. The `JButton` has to be pushed to make something happen. It consists of a label and/or an icon that describes its function, an empty area around the text/icon and a border. By default, the border is a special border that reflects the status of the button.

**A `JButton` can be constructed by any of the constructors mentioned below:**

- `JButton()` : Creates a button with no text or icon
- `JButton(Icon icon)` : Creates a button with icon.
- `JButton(String text)` : Creates a button with text
- `JButton(String text, Icon icon)` : Creates a button with text and icon

**Some methods can be used in conjunction with a `JButton` are listed below:**

-

isDefaultButton() : Returns whether or not the corresponding button is the default button on the RootPane.

- isDefaultCapable() : Returns whether or not the corresponding button is capable of being the default button on the RootPane.
- setDefaultCapable(boolean defaultCapable) : Sets whether or not the corresponding button is capable of being the default button on the RootPane.

### **Right-clicks on a Button**

The default action of a JButton is to receive a left mouse click. The button could be programmed to receive a right mouse click also. There are ways in which this can be achieved.

- Creating our own UI for JButton
- Overlay the button with an invisible component that would intercept all events and pass through all except right clicks.
- Subclass JButton and override the process MouseEvent() method

**JCheckBox:** A JCheckBox is a control that may be turned on and off by the user to designate some kind of property being selected or not selected. It consists of a background rectangle, and a text string and/or icon. The JCheckBox normally shows its current state visually. This is done by placing a check mark in a box, or by changing the icon.

A JCheckBox generates item events when its state changes. The checkbox can be created by using any one of the constructors mentioned below:

- JCheckBox() : Creates an initially unchecked checkbox with no text or icon.
- JCheckBox(Icon icon) : Creates an initially unchecked checkbox with an icon.  
JCheckBox(Icon icon, Boolean selected) : Creates a checkbox with an icon and specifies whether or not it is initially selected
- JCheckBox(String text) : Creates an initially unchecked checkbox with the specified text.  
JCheckBox(String text, Boolean selected) : Creates a checkbox with the specified text and specifies whether or not it is initially selected.
- JCheckBox(String text, Icon icon) : Creates an initially unselected checkbox with the text and icon specified.
- JCheckBox(String text, Icon icon, Boolean selected) : Creates a checkbox with icon and text and specifies whether or not it is initially selected.

**JRadioButton:** This is normally used as one of a group of radio buttons of which only one may be selected at a time. These are grouped using a ButtonGroup and are usually used to select from a set of mutually exclusive options. It consists of a background rectangle and text and/or an icon. If it includes an icon, the icon is used to visually reflect the current state of the radio button.

**Using the constructors listed below , radio buttons can be created:**

- JRadioButton() : Creates an initially unselected radio button with no set text.

- `JRadioButton(Icon icon)` : Creates an initially unselected radio button with the specified image but no text.
- `JRadioButton(Icon icon, Boolean selected)` : Creates a radio button with the specified image and selection state, but no text.
- `JRadioButton(String text)` : Creates an initially unselected radio button with the specified text.
- `JRadioButton(String text, boolean selected)` : Creates a radio button with specified text and selection state.
- `JRadioButton(String text, Icon icon)` : Creates a radio button that has the specified text and image, and that is initially unselected.
- `JRadioButton(String text, Icon icon, boolean selected)` : Creates a radio button that has the specified text, image, and selection state.

## Programs:

Followig is the programm to display an Applet.

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
/*

<applet code = "Applets.class" width = 250 height = 250
> </applet>

*/
public class Applets extends JApplet
{

    JButton B1;
    public void init()
    {

        JPanel contentpane = (JPanel) getContentPane();
        B1= new JButton("My First Applet");
        contentpane.add(B1);
```

```
}  
}
```

- Save the file as **Applets. Java**
- 
- Compile the file using **javac Applet.java**

On successful compilation, execute the file using **appletviewer Applets.java**

The output appers as shown in following figure :

GCEER



**The following program is an example of  
Jframe/JButton**

```
import  
java.awt.*;  
import
```

```
java.awt.event.*
; import
javax.swing.*;

public class Button1 extends JFrame implements
ActionListener

{

    JButton
    mtextbtn1;
    JButton
    mtextbtn2;

    public Button1()
    {
        setTitle("Button Example");

        JPanel contentpane =
        (JPanel) getContentPane();
        contentpane.setLayout(new
        GridLayout(2,2));

        mtextbtn1= new
        JButton("Enabled");
        mtextbtn1.setMnemonic('E');
        mtextbtn1.addActionListener(
        this);
        contentpane.add(mtextbtn1);

        mtextbtn2 = new
        JButton("Disabled");
        mtextbtn2.setMnemonic('D');
        mtextbtn2.addActionListener(t
```

```
his);
contentpane.add(mtextbtn2);

mtextbtn1.setEnabled(true);

myadapter myapp = new
myadapter();
addWindowListener(myapp);
}

class myadapter extends WindowAdapter
{
    public void windowclosing(WindowEvent e)
    {
        System.exit(0);
    }
}

public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == mtextbtn1)
    {
        setTitle("First button clicked");
    }
    else if ( e.getSource() == mtextbtn2)
    {
        setTitle("Second button clicked");
    }
}
```

```

    }
}

public static void main(String args[])
{

    Button1 b = new
    Button1();
    b.setSize(100,100);
    b.setVisible(true);

}
}

```

- Save the file as **Button1.java**
- Compile the program using **javac**  
**Button1.java** Execute the program using **java**
- **Button1**

The output appears as shown in following figure.

#### Example program for JCheckBoxes/JFrame.

```

import
java.awt.*;
import
java.awt.event.*
; import
javax.swing.*;

```



```
public class checkbox1 extends JFrame implements
ItemListener
{

    JCheckBox
checkbox;
public
checkbox1()
{

    setTitle("Check box Example");

    JPanel contentpane =
(JPanel) getContentPane();
contentpane.setLayout(new
GridLayout(2,2)); checkbox = new
JCheckBox("Toggle");
checkbox.addItemListener(this);
contentpane.add(checkbox);

    myadapter myapp = new
myadapter();
addWindowListener(myapp);
}

class myadapter extends WindowAdapter
{

    public void windowclosing(WindowEvent e)
    {

        System.exit(0);

    }

}
```

```
}

public void itemStateChanged(ItemEvent e)
{

if (e.getStateChange() == ItemEvent.SELECTED)
{

    setTitle("Checkbox selected");
}

else
{

    setTitle("Checkbox unselected");
}

}

}

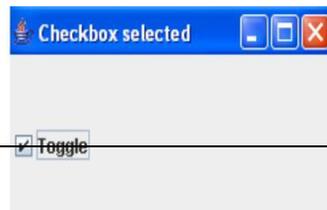
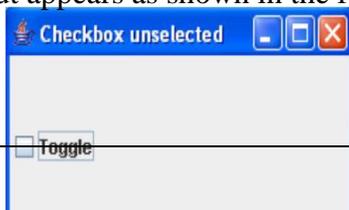
public static void main(String args[])
{

    checkbox1 c = new
checkbox1(); c.setSize(250,250);
c.setVisible(true);
}

}
```

- Save the file as **checkbox1.java**
- 
- Compile the file using javac **checkbox1.java**
- 
- Execute the file using **java checkbox**

The output appears as shown in the following figure



## Example program for JRadioButtons

```
import java.awt.*; import java.awt.event.*;
import javax.swing.*;

public class Radiobuttons extends JFrame implements ItemListener

{
    JRadioButton rb1, rb2;
    ButtonGroup grp = new ButtonGroup();

    public Radiobuttons()
    {
        setTitle("Radio Buttons Example");

        JPanel contentpane = (JPanel) getContentPane();
        contentpane.setLayout(new FlowLayout());

        rb1 = new JRadioButton("Enabled");
        rb1.addItemListener(this);
        rb1.setEnabled(true); contentpane.add(rb1);

        rb2 = new JRadioButton("Disabled");

        rb2.addItemListener(this); //rb2.setActionCommand("Two Activated");
        contentpane.add(rb2);
        rb2.setEnabled(false);

        grp.add(rb1);
        grp.add(rb2);

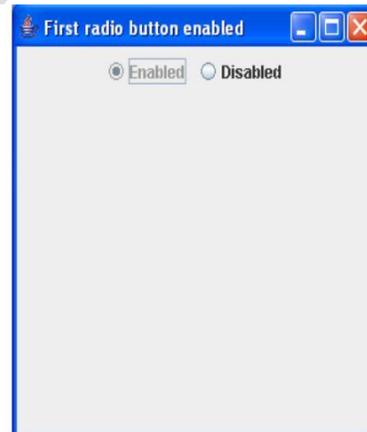
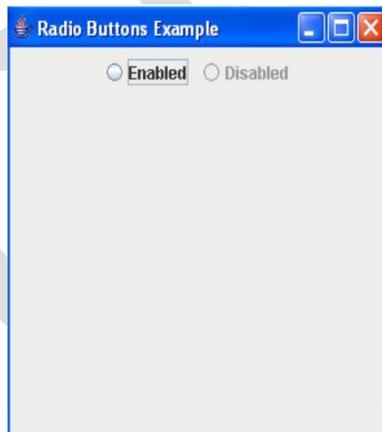
        myadapter myapp = new myadapter();
        addWindowListener(myapp);
    }
}
```

```
    }  
class myadapter extends WindowAdapter  
{  
  
    public void windowClosing(WindowEvent e)  
    {  
  
        System.exit(0);  
    }  
}  
  
public void itemStateChanged(ItemEvent e)  
{  
  
    if (e.getSource()==rb1)  
    {  
  
        setTitle("First radio button enabled");  
        rb1.setEnabled(false);  
        rb2.setEnabled(true);  
  
    }  
    else if(e.getSource()==rb2)  
    {  
  
        setTitle("Second radio button enabled");  
        rb1.setEnabled(true); rb2.setEnabled(false);  
  
    }  
  
}  
  
public static void main(String args[])
```

```
{  
  
Radiobuttons rb = new Radiobuttons();  
rb.setSize(300,300); rb.setVisible(true);  
}  
}
```

- Save the file as **Radiobuttons.java**
- Compile the file using **javac Radiobuttons.java**
- On successful compilation execute the file using **java Radiobuttons**

The output appears as shown in the following figure :



## JDBC ARCHITECTURE

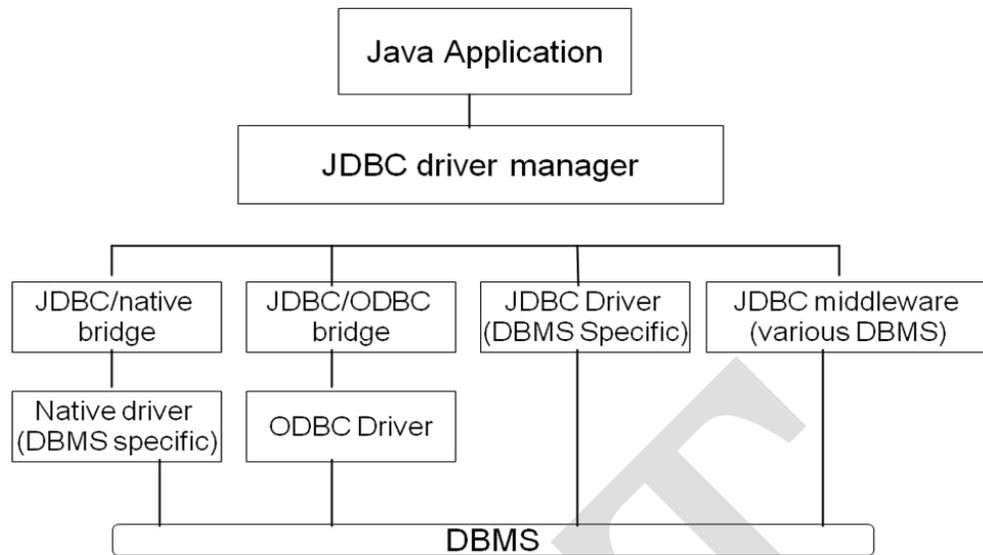
### 14.1

#### Introduction to JDBC

JDBC stands for Java Database Connectivity. It is set of Java API's(application programming interface) used for executing SQL statements. This API consists of a set of classes and interfaces to enable programmers to write pure Java Database applications.

JDBC is a software layer that allows developers to write real client –server projects in Java. JDBC does not concern itself with specific DBMS functions. JDBC API defines how an application opens a connection, communicates with a database, executes SQL statements, and retrieves query result. Following fig. will illustrate the role of JDBC. JDBC is based on the X/OPEN call level interface (CLI) for SQL.

Call Level Interface is a library of function calls that supports SQL statements. CLI requires neither host variables nor other embedded SQL concepts that would make it less flexible from a programmer's perspective. It is still possible, however, to maintain and use specific functions of a database management system when accessing the database through a CLI.



JDBC was designed to be very compact, simple interface focusing on the execution of raw SQL statements and retrieving the results. The goal of creating JDBC is to create an interface that keeps simple tasks, while ensuring the more difficult and uncommon tasks are at least made possible.

**The following are the characteristics of JDBC.**

- It is a call-level SQL interface for java
- 
- It does not restrict the type of queries passed to an underlying DBMS driver
-

JDBC mechanism are simple to understand and use

It provides a java interface that stays consistent with the rest of the Java system

JDBC may be implemented on top of common SQL level APIs.

Microsoft ODBC API offers connectivity to almost all databases on all platforms and is the most widely used programming interface for accessing relational database. But ODBC cannot be directly used with java programs due to various reasons enumerated in the JDBC vs. ODBC section. Hence the need for JDBC came into existence.

It is possible to access various relational databases like Sybase, Oracle, Informix, Ingers, using JDBC API. Using JDBC, we can write individual programs to connect to individual database or one program that take care of connecting to the respective database.

## **14.2**

### **Java and JDBC**

The combination of java with JDBC is very useful because it lets the programmer run his/her program on different platforms, Java programs are secure, robust, automatically downloaded from the network and java is a good language to create database applications. JDBC API enables Java applications to interact with different types of database. It is possible to publish vital information from a remote database on a web page using a java applet. With increasing inclination of programmers towards Java, knowledge about JDBC is essential.

**Some of the advantages of using Java with JDBC are as follows:**

- Easy and economical
- 
- Continued usage of already installed databases

Development time is short

Installation and version control simplified

**How does JDBC work**

JDBC defines a set of API objects and methods to interact with the underlying database. A Java program first opens a connection to the database, makes a statement object, passes SQL statements to the underlying database management system (DBMS) through the statement object and retrieve the results as well as information about the result set.

There are two types of interfaces – low –level interface and high-level interface. While high level interfaces are user-friendly, low-level interfaces are not. JDBC is a low-level API

interface, ie. it used to invoke or call SQL commands directly. The required SQL statements are passed as strings to Java methods.

Some of the current trend that are being developed to add more features to JDBC are embedded SQL for java and direct mapping of relational database to java classes.

Embedded SQL enables mixing of java into SQL statements. These statements are translated into JDBC calls using SQL Processor. In this type of direct mapping of relational database tables to java, each row of the table becomes an instance of the class and each column value corresponds to an attribute of that instance. Mapping is being provided that makes rows of multiple tables to form a java class.

### **14.3 JDBC VS ODBC**

The most widely used interface to access relational database today is Microsoft's ODBC

API. ODBC performs similar tasks as that of JDC(Java Development Connection) and yet JDBC is preferred due to the following reasons :

- ODBC cannot be directly used with Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness and automatic portability of applications.
- 
- ODBC makes use of pointers which have been totally removed from Java

ODBC mixes simple and advanced features together and has complex options for simple queries. But JDBC is designed to keep things simple while allowing advanced capabilities when required.

- JDBC API is a natural Java Interface and is built on ODBC. JDBC retains some of the basic features of ODBC like X/Open SQL Call Level Interface.
- JDBC is to Java programs and ODBC is to programs written in languages other than Java.
- ODBC is used between applications and JDBC is used by Java programmers to connect to databases.

### **Details about JDBC**

The JDBC API is in the package java.sql it consists of 8 interfaces, 6 classes and 3 exceptions in JDK1.1.

Interfaces:

- CallableStatement
- 
- 
- 
- 
- 
- 
-

Connection

DatabaseMetaData

Driver

PreparedStatement

ResultSet

ResultSetMetaData

Statement

Classes:

- 
- 
- 
- 
- 
- 

Date

DriverManager

DriverPropertyInfo

Time

Timestamp

Types

Exceptions:

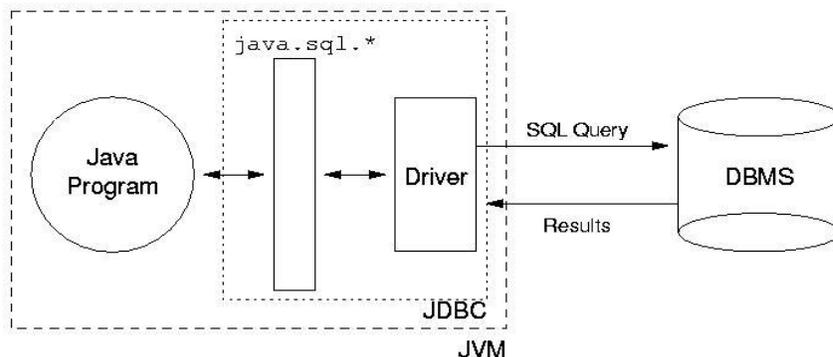
- •
- DataTruncation
- 

SQLException

SQLWarning

## 14.4

### JDBC DRIVER MODEL



### JDBC Driver Types

There are 4 types of JDBC drivers. Commonest and most efficient of which are type 4 drivers. Here is the description of each of them:

- **JDBC Type 1 Driver** - They are JDBC-ODBC Bridge drivers ie. Translate JDBC into ODBC and use Windows ODBC built in drivers. They delegate the work of data access to ODBC API. They are the slowest of all. SUN provides a JDBC/ODBC driver implementation.

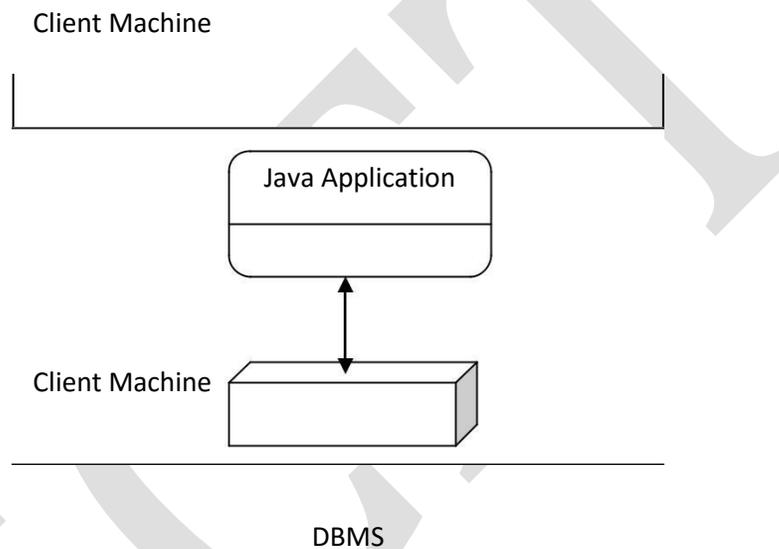
**JDBC Type 2 Driver** - They mainly use native API for data access ie. Converts JDBC to data base vendors native SQL calls and provide Java wrapper classes to be able to be invoked using JDBC drivers like Type 1 drivers; requires installation of binaries on each client.

- **JDBC Type 3 Driver** - Translates JDBC to a DBMS independent network protocol. They are written in 100% Java and use vendor independent Net-protocol to access a vendor independent

- remote listener. This listener in turn maps the vendor independent calls to vendor dependent ones. This extra step adds complexity and decreases the data access efficiency.
- **JDBC Type 4 Driver** - They are also written in 100% Java and are the most efficient among all driver types. It compiles into the application, applet or servlet; doesn't require anything to be installed on client machine, except JVM. It also converts JDBC directly to native API used by the RDBMS.

The JDBC API supports both two-tier and three-tier processing models for database access.

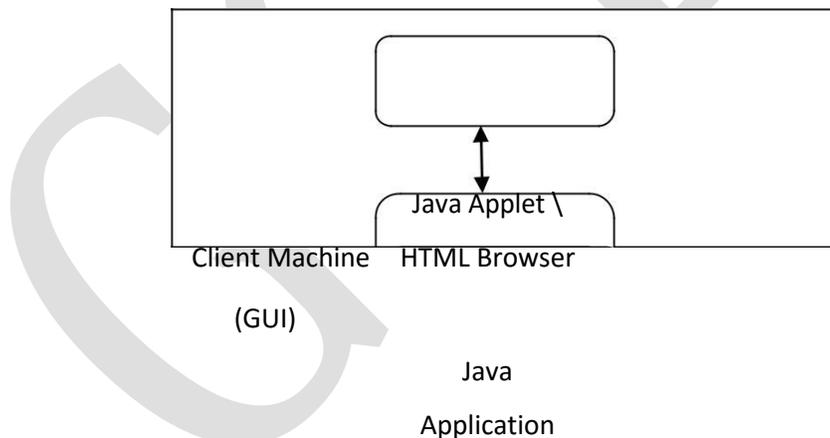
### Two-tier Architecture for Data Access



In the two-tier model, a Java application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

In the three-tier model, commands are sent to a "middle tier" of services, which then sends the commands to the data source. The data source processes the commands and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide performance advantages.

### Three-tier Architecture for Data Access



until recently, the middle tier has often been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code and technologies such as Enterprise JavaBeans™, the Java platform is fast becoming the standard platform for middle-tier development. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features.

- With enterprises increasingly using the Java programming language for writing server code, the JDBC API is being used more and more in the middle tier of a three-tier architecture. Some of the features that make JDBC a server technology are its support for connection pooling, distributed transactions, and disconnected row sets. The JDBC API is also what allows access to a data source from a Java middle tier.

## **SQL CONFORMANCE**

Structured Query Language (SQL) is the standard language used to access relational databases, unfortunately, there are no standards set at present for it for ex, problems may arise due to the variations in different data types of different databases. JDBC defines a set of generic SWL types identifiers in the class `Java.SQL.Types`

### **Ways of dealing with SQL conformance**

**JDBC deals with SQL conformance by performing the following :**

- JDBC API allows any query string to be passed through to an underlying DBMS driver. But there are possibilities of getting an error on some DBMS.
- Provision of ODBC style escape closes.
- Provision of descriptive information about the DBMS using an interface, `DatabaseMetaData`.

The designation JDBC Compliant was created to set a standard level of JDBC functionality on which users can rely. Only the ANSI SQL 2 entry level supported drivers can make use of this designation. The conformance tests check for the existence of all classes and methods defined in the JDBC API and SQL entry level functionality.

## **Types of Driver Managers**

JDBC contains three components: Application, Driver Manager, Driver. The user application invokes JDBC methods to send SQL statements to the database and retrieves results. JDBC driver manager is used to connect Java applications to the correct JDBC driver. JDBC driver test suite is used to ensure that the installed JDBC driver is JDBC Compliant. There are four different types of JDBC drivers as follows

### **1.The JDBC-ODBC Bridge plus ODBC driver :**

The JDBC-ODBC Bridge plus ODBC driver is a JavaSoft Bridge protocol that provides JDBC access via ODBC drivers. But as we have mentioned earlier, combining ODBC brings in a lot of drawbacks and limitations, since the ODBC driver has to be installed on each client machine, it is not advisable to choose this type of driver for large networks.

### **2. Native-API partly-Java driver :**

Native-API partly-Java driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix or other DBMS. But some binary code has to be loaded on all client like the bridge driver and hence is not suitable for large networks.

### **3.JDBC-Net pure Java driver:**

JDBC-Net pure Java driver translates JDBC calls into DBMS independent net protocol. A server again translates this protocol to a DBMS protocol. The net server middleware connects its pure Java clients to many different databases. The type of protocol in this middleware depends on the vendor.

#### **4. Native-protocol pure Java driver :**

Native-protocol pure Java driver convert JDBC calls to network protocols used by the DBMSs directly. Requests from client machines are made directly to the DBMS server.

Drivers 3 and 4 are the most preferred ways to access databases from JDBC drivers.

## **Database connectivity**

### **Introduction :**

A Database connection is a facility in computer science that allows client software to communicate with database server software, whether on the same machine or not. A connection is required to send commands and receive answers.

Connections are built by supplying an underlying driver or provider with a connection string, which is a way of addressing a specific database or server and instance as well as user authentication credentials (for example, *Server=sql\_box;Database=Common;User ID=uid;Pwd=password;*). Once a connection has been built it can be opened and closed at will, and properties (such as the command time-out length, or transaction, if one exists) can be set. The Connection String is composed of a set of key/value pairs as dictated by the data access interface and data provider being used.

### **15.2**

#### **A connection can be open with the help of following steps**

1. Importing Packages
2. Registering the JDBC Drivers
3. Opening a Connection to a Database
4. Creating a Statement Object
5. Executing a Query and Returning a Result Set Object

6. Processing the Result Set
7. Closing the Result Set and Statement Objects
8. Closing the Connection

### **Step 1. Importing Packages**

The following JDBC packages will be imported for creating connection. java.sql.

java.math.

java.io.

oracle.jdbc.driver.

### **Step 2. Registering the JDBC Drivers**

Following four parameters are required to register JDBC

Drivers. ○ Database URL

○ JDBC Driver

name ○ User Name

○ Password

JDBC Drivers can be register using following

methods. ○ Class

```
drvClass=Class.forName(m_driverName);
```

```
○ DriverManager.registerDriver((Driver)drvClass.newInstance());
```

### **Step 3 : Opening a Connection to a Database**

Connection to the underlying database can be opened using

Connection

: For simple SQL statements ( no parameter )

m\_con=DriverManager.getConnection(m\_url,m\_userName,m\_password);

: For SQL statements with one or more IN parameters, or simple

For executing SQL stored procedures.

### **Step 4 : Creating a Statement Object**

#### **SQL Statements**

Once a connection is established, It is used to pass SQL statements to its underlying database. JDBC provides three classes for sending SQL Statements to the database, where PreparedStatement extends from Statement, and CallableStatement extends from

PreparedStatement

nt: ○

### Statement

- **PreparedStatement**
- **CallableStatement**

The statement interface provides three different methods for **executing SQL statements** :

- **executeQuery** : For statements that produce a single result set.
- **executeUpdate** : For executing **INSERT**, **UPDATE**, or **DELETE** statements and also **SQL DDL** (Data Definition Language) statements.
- **execute** : For executing statements that return more than one result set, more than one update count, or a combination of the two.

A Statement object is used with following steps:

### Statement

```
Statement stmt=m_con.createStatement();
```

```
Statement stmt=m_con.createStatement(int resultSetType, int resultSetConcurrency);
```

### PreparedStatement

```
PreparedStatement pstmt=m_con.prepareStatement(String sql);
```

```
PreparedStatement pstmt=m_con.prepareStatement(String sql, int resultSetType, int  
resultSetConcurrency),
```

Note:

The SQL parameter could contain one or more `_?` in it. Before a `PreparedStatement` object is executed, the value of each `_?` parameter must be set by calling a `setXXX` method, where `XXX` stands for appropriate type for the parameter. For ex. If the parameter has a java type of `String`, the method to use is `setString`.

### **CallableStatement**

```
CallableStatement csmt=m_con.prepareCall(String sql);
```

```
CallableStatement csmt=m_con.prepareCall(String sql, int resultSetType, int  
resultSetConcurrency,);
```

Note :

The sql parameter is in the form of `{call <stored_procedure_name>[(arg1, arg2,...)]}` or `{?=call <stored_procedure_name>[(arg1,arg2...)]}`. It could contain one or more `_?`'s in it, which indicates IN, OUT or INOUT parameters. The value of each IN parameter is set by calling a `setXXX` method, while each OUT parameter should be registered by calling a `registerOutParameter` method.

### **Step 5: Executing a Query and Returning a Result Set Object AND**

### **Step 6: Processing the Result set**

#### **Execute the Statement**

**Statement :**

```
ResultSet res=stmt.executeQuery(String sql);  
int rowCount=stmt.executeUpdate(String sql);  
boolean result=stmt.execute(String sql);
```

### **PrepaedStatement :**

```
ResultSet res=pstmt.executeQuery();  
int rowCount=pstmt.executeUpdate();  
boolean result=pstmt.execute();
```

### **CallableStatement :**

```
ResultSet res=cstmt.executeQuery(); int  
rowCount=cstmt.executeUpdate();  
boolean result=cstmt.execute();
```

### **Processing the Result set**

A result set contains all of the rows which satisfied the conditions in an SQL statement and it provides access to the data in those rows through getXXX mehods that allow access to the various columns of the current row.

The ResultSet.next() method is used to move to the next row of the ResultSet, making the next row become the current row. ResultSet.next() returns true if the new current row is valid, false if there are no more rows. After all the works have been done, the ResultSet should be closed with ResultSet.close() method.

Because of limitations imposed by some DBMSs, it is recommended that for maximum portability, all of the results generated by the execution of a CallableStatement object should be retrieved before OUT parameters are retrieved using CallableStatement.getXXX methods.

### **Step 7: Closing the Result Set and Statement Objects**

## Close the statement

After all the works have been done, the result set and statement should be closed with the following code :

```
Resultset          : rset.close();  
Statement          : stmt.close();  
PrepaedStatement  : pstmt.close();  
CallableStatement : cstmt.close();
```

## Step 8: Closing the Connection

After all the works have been done, the Connection should be closed with the following code:

```
(Connection name)m_con.close();
```

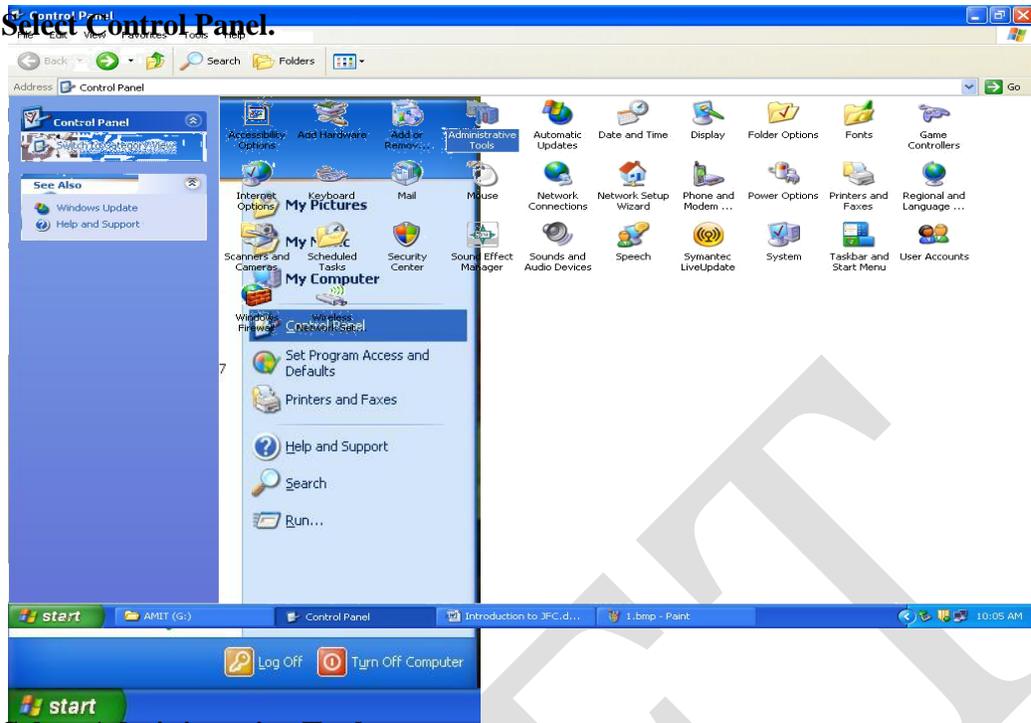
## 15.3

### Connecting to an ODBC Data Source

A database can be created and managed through Java applications. Java application that uses a JDBC-ODBC bridge to connect to a database file either a dbase, Excel, FoxPro, Access, SQL Server, Oracle or any other. Open the ODBC Data source from the control panel. A database can be created and managed through Java applications.

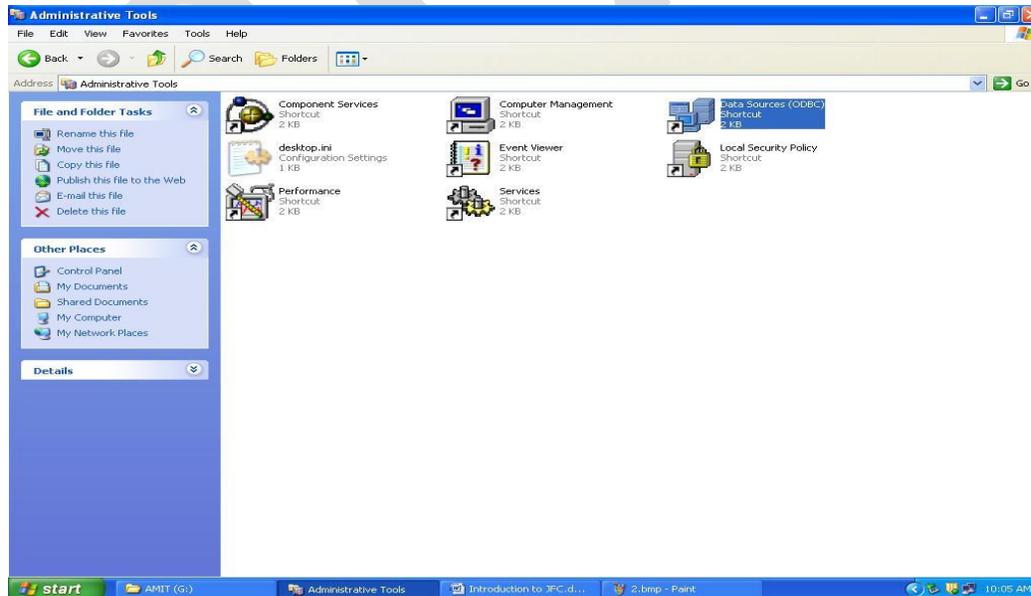
Follow the following steps to connect to an ODBC Data Source for —ORACLE|.

### 1. Select Control Panel.

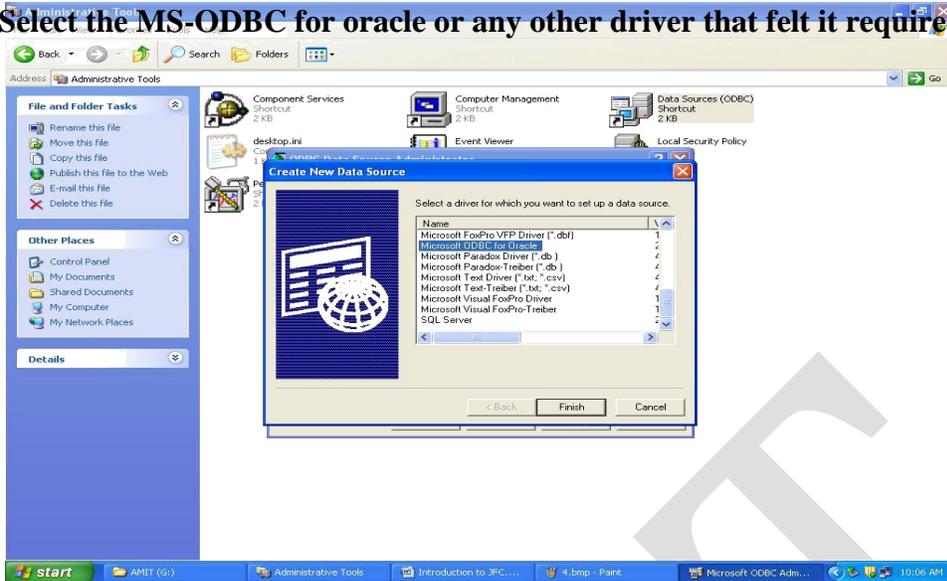


### Select Administrative Tool

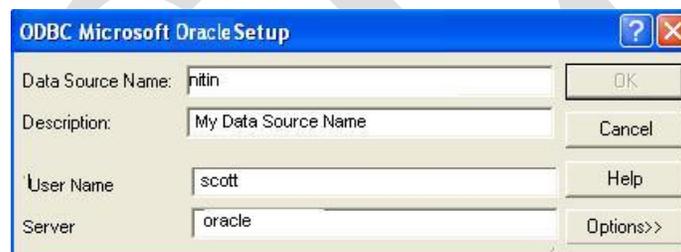
### 3. Select "Data Sources (ODBC)" icon



**4. Select the MS-ODBC for oracle or any other driver that felt it required.**

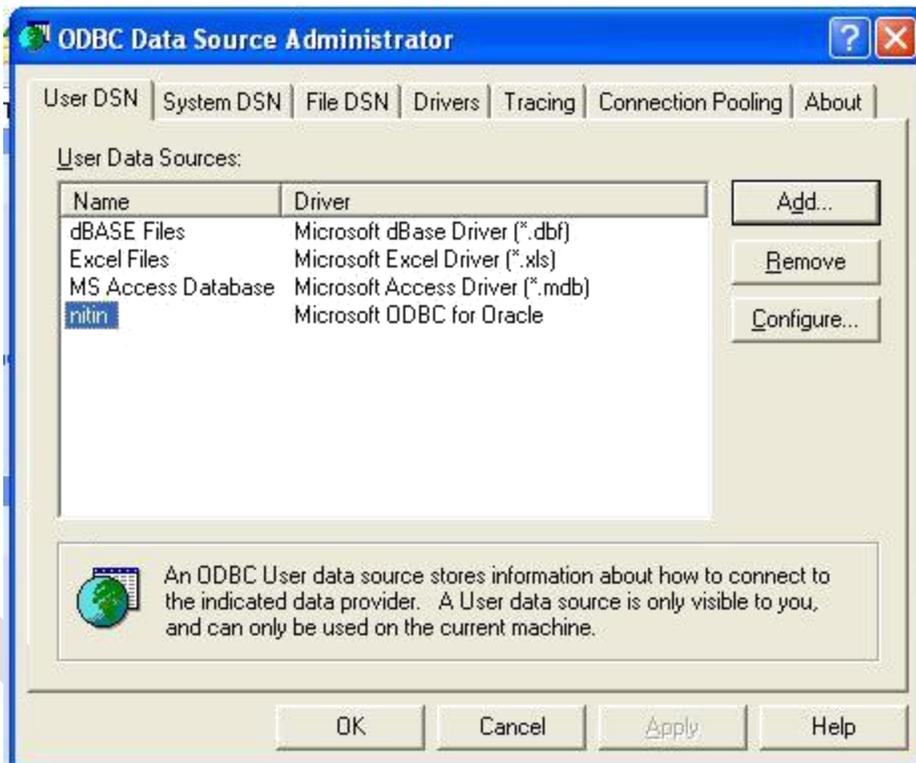


**5. Once clicking the finish button, the following window appears asking for Data Source name, description etc.**



**6. Provide “Data Source Name”, “Discription”,”Username” and “Server” name. The username and Server name can be obtained from the Administrator. Click on ok button.**

The DSN is now ready and the Java code can be written to access the database’s tables.



## 15.4

### JDBC Programs

#### 1. Example for creating Table.

// Create Table

```
import java.sql.*; // imports all classes that belongs to the package java.sql.*
```

```
public class CreateTab
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        try
```

```
        {
```

```
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
            Connection con= DriverManager.getConnection  
                (jdbc:odbc:nitinl ,scottl,ltigerl);
```

```
            // specifies the type of driver as JdbcOdbcDriver. Statement stat=  
            con.createStatement();
```

```
                String str="Create table T1(Rno number(2), Stdname varchar2(20))";
```

```
            Stat.executeUpdate(str);
```

```
            System.out.println("Table created successfully");
```

```
        }
```

```
    catch(SQLException e 1)
```

```
    {
```

```
        System.out.println("Errors" + e 1);
```

```
    }
```

```
Catch(ClassNotFoundException e 2)
{
    System.out.println("Errors|| + e 2);
}
```

## 2.Example for inserting records into a Table

```
// Insert into table
```

```
import java.sql.*;
```

```
public class InsertTab
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        ResultSet result;
```

```
        try
```

```
        {
```

```
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
            Connection con= DriverManager.getConnection
```

```
                ("jdbc:odbc:nitinl ,scottl,ltigerl");
```

```
            Statement stat= con.createStatement();
```

```
                Stat.executeUpdate("Insert into T1 values(20,'Smith')");
```

```
Stat.executeUpdate(—Insert into T1 values(21,‘John‘)l);
Stat.executeUpdate(—Insert into T1 values(22,‘Kate‘)l);
```

```
Stat.executeUpdate(—Insert into T1 values(23,‘Stive‘)l);
```

```
System.out.println(Rows Inserted successfullyl);
```

```
result=stat.executeQuery(—Select * from T1l);
```

```
while(result.next())
```

```
{
```

```
System.out.println(result.getInt(1)+result.getString(2));
```

```
}
```

```
}
```

```
catch(Exception e)
```

```
{
```

```
System.out.println(—Errorsl+e);
```

```
}
```

```
}
```

```
}
```

### 3.Example for viewing rows from a table

```
// viwing from emp table
```

```
import java.sql.*; public class SelectEmp
```

```
{
```

```
public stativ void main(String args[])
```

```
{
```

```
String url=|jdbc:odbc:nitin|;
```

```
Connection con;
```

```
String s= —select ename from emp 1||;
Statement stmt;

try
{
    Class.forName(—sun.jdbc.odbc.JdbcOdbcDriver||);
}
catch(java.lang.ClassNotFoundException e)
{
    System.err.println(—ClassNotFoundException:|);
    System.err.println(e.getMessage());
}

try
{
    con=DriverManager.getConnection(url,|Scott|,|Tiger|);

    stmt=con.createStatement(); resultSet rs=stmt.executeQuery(s); while(rs.next())
    {
        String s1=rs.getString(—ename|); System.out.println(—Employee name:| +s1);
    }
    stmt.close();

    con.close();
}
catch(SQLException ex)
{
    System.err.println(—SQLException:|+ex.getMessage());
}
```

```
}  
}
```

#### 4. Example using prepared statements

```
import java.sql.*;  
  
public class PreStExample  
{  
    public static void main(String[] args)  
    {  
        Connection con = null; PreparedStatement prest; try{  
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
  
            Connection con= DriverManager.getConnection("jdbc:odbc:nitinl ,scottl,tigerl"); try{  
  
                String sql = "SELECT stdname FROM T1 WHERE Rno = ?"; prest =  
                con.prepareStatement(sql);  
  
                prest.setInt(1,21);  
  
                ResultSet rs1 = prest.executeQuery(); while (rs1.next())  
  
                {  
                    String stname = rs1.getString(1);  
  
                    System.out.println("student name is: "+stname);  
                }  
                prest.setInt(1,23);  
                ResultSet rs2 = prest.executeQuery();
```

```
while (rs2.next())
{
String stname1 = rs2.getString(1);

    System.out.println("student name is: "+stname1);
}
}

catch (SQLException s){
    System.out.println("SQL statement is not executed!");
}

}

catch (Exception e){ e.printStackTrace();
}
}
```

## **Features of java**

### **15 Additional topics**

No additional topics

### **16 University Question papers of previous years**

GCEET

Code No: 114CX

R13

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech II Year II Semester Examinations, May-2015

JAVA PROGRAMMING

(Common to CSE, IT)

Time: 3 Hours

Max. Marks: 75

**Note:** This question paper contains two parts A and B.  
Part A is compulsory which carries 25 marks. Answer all questions in Part A.  
Part B consists of 5 Units. Answer any one full question from each unit.  
Each question carries 10 marks and may have a, b, c as sub questions.

**Part- A**

(25 Marks)

- 1.a) What is data abstraction? [2M]
- b) List string manipulation functions of Java String class. [3M]
- c) Differentiate between interface and abstract class. [2M]
- d) Explain the use of 'final' keyword. [3M]
- e) Differentiate between thread and process. [2M]
- f) List any six built-in exceptions in Java. [3M]
- g) What is the difference between array and vector? [2M]
- h) List the byte stream classes. [3M]
- i) What are the containers available in swing? [2M]
- j) Compare Applets with application programs. [3M]

**Part-B**

(50 Marks)

- 2.a) Explain the basic concepts of object oriented programming.
  - b) What is the usage of enumerated data type? Give examples. [5+5]
- OR**
- 3.a) Discuss Java jump statements.
  - b) Write about garbage collection in Java.
  - c) Explain the use of 'this' keyword. [3+3+4]
- 4.a) Explain method overriding with a suitable example program.
  - b) With suitable program segments describe the usage of 'super' keyword. [5+5]
- OR**
- 5.a) What is a nested class? Differentiate between static nested classes and non-static nested classes.
  - b) How to define a package? How to access, import a package? Explain with examples. [5+5]
- 6.a) With a suitable Java program explain user-defined exception handling.
  - b) What is meant by re-throwing exception? Discuss a suitable scenario for this. [5+5]
- OR**
- 7.a) Does Java support thread priorities? Justify your answer with suitable discussion.
  - b) Describe producer-consumer pattern using inter-thread communication. [5+5]

- 8.a) Give an account of Random collection class
- b) Discuss the methods of Stack class
- c) What is the need of Generics?

[3+3+4]

**OR**

- 9.a) Discuss the four types of JDBC driver with suitable diagrams.
- b) Write a JDBC program to update the amount balance in an account after every withdrawal. Assume the necessary database table.

[5+5]

- 10.a) What is the significance of layout managers? Discuss briefly various layout managers.
- b) Give an overview of JButton class.

[5+5]

**OR**

- 11.a) Explain delegation event model.
- b) Write an Applet to draw a smiley picture accept user name as a parameter and display welcome message.

[5+5]

--ooOoo--

p. 05.07.14  
cse - 21  
Code No: 09A40503

R09

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD**  
B.Tech II Year II Semester Examinations, June-2014  
**OBJECT ORIENTED PROGRAMMING**  
(Common to CSE, IT)

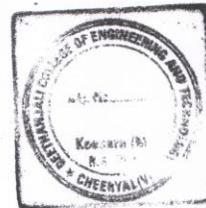
Time: 3 hours

Max. Marks: 75

Answer any five questions  
All questions carry equal marks  
---

- 1.a) Write in detail on interfaces and abstract classes in Java.  
b) Write in detail on the features of OO programming.
- 2.a) Write a static recursive method for returning the sum of digits of an integer.  
b) Write in detail on methods of String class.
- 3.a) Write in detail on member access rules.  
b) Write a Java program to print the sum of the numbers that are supplied as command line arguments.
- 4.a) Write a Java program to reverse the contents of a text file.  
b) Write briefly on Reader and Writer classes.
- 5.a) Write a Java program to read a text file and print the number of unique words.  
b) Write briefly on Exception handling.
- 6.a) Write in detail on annotations with examples.  
b) Write a Java program to create multiple threads.
7. Write a Java program to implement an AWT based calculator with basic operations.
- 8.a) Write briefly on event sources, event classes and event listeners in Java.  
b) Write briefly on Adapter classes.

\*\*\*\*\*



36

R09

Code No: 09A40503  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD  
B.Tech II Year II Semester Examinations, November/December 2013  
OBJECT ORIENTED PROGRAMMING  
(Common to CSE, IT)

Max. Marks: 75

Time: 3 hours

Answer any five questions  
All questions carry equal marks  
---

- 1.a) What is a class and object? Is there any relationship between them? Explain. [8+7]  
b) What is a member function and data members? Explain briefly.
- 2.a) What is the order of invoking constructors for the classes that make up the hierarchy? Explain. [8+7]  
b) Discuss the need for overriding methods.
- 3.a) Define multiple inheritance. Does Java Support multiple inheritance? Justify your answer. [8+7]  
b) Write java program to implement the multilevel Inheritance.
- 4.a) Discuss in detail about nested Interfaces. [8+7]  
b) What happens when an Interface is partially implemented? Explain.
- 5.a) Discuss with a sample Java program explaining the need of defining multiple catch clauses. [8+7]  
b) What is meant by nested try statements? When will they be used? Explain it with a sample Java program.
6. What are the various methods defined in the Thread class? Explain their usage with a sample Java program. [15]
- 7.a) What are the advantages of Layout managers? Explain the different Layout Managers AWT supports. [8+7]  
b) What is preferred size of a component and how it is related to the Layout Managers?
- 8.a) Discuss various constructors and methods that are defined in the JComboBox class and JComponent class. [8+7]  
b) Write a sample Java program to show how to create Combo boxes and labels.

\*\*\*\*\*

Code No: R05220402

Set No. 1

## II B.Tech II Semester Regular Examinations, Apr/May 2008

### OOP THROUGH JAVA

( Common to Electronics & Communication Engineering, Computer Science & Engineering, Information Technology, Computer Science & Systems Engineering, Electronics & Telematics and Electronics & Computer Engineering)

Time: 3 hours Max Marks: 80

Answer any FIVE Questions

All Questions carry equal marks

?????

1. (a) Describe the genesis of java. Also write brief overview of java  
(b) Write a program that will read an unspecified number of integers and will determine how many positive and negative values have been read. Your program ends when the input is 0. [8+8]
2. (a) What is a constructor? What are its special properties?  
(b) How do we invoke a constructor?  
(c) What are objects? How are they created from a class? [6+4+6]
3. (a) Explain about final classes, final methods and final variables?  
(b) Explain about the abstract class with example program? [8+8]
4. Prove that the fields in an interface are implicitly static and final. [16]
5. (a) Why thread is called light weight task and process heavy weight task.  
(b) What are the different things shared by different threads of a single process. What are the benefits of sharing?  
(c) Is multithreading suitable for all types of applications. If yes explain any such application. If no, explain any application for which multithreading is not desired.  
[4+4+8]
6. What are the methods supported by KeyListener interface and MouseListener interface. Explain each of them with examples. [8+8]
7. Explain the functionality of JComponent with example. Differentiate JComponent and JPanel. [8+8]
8. (a) What are accessor methods?  
(b) How will you create strings and stringbuffers? How will you modify them?

**Code No: R05220402**

**Set No. 2**

**II B.Tech II Semester Regular Examinations, Apr/May 2008**

**OOP THROUGH JAVA**

( Common to Electronics & Communication Engineering, Computer Science & Engineering, Information Technology, Computer Science & Systems Engineering, Electronics & Telematics and Electronics & Computer Engineering)

Time: 3 hours Max Marks: 80

Answer any FIVE Questions

All Questions carry equal marks

1. (a) Describe the genesis of java. Also write brief overview of java.  
(b) Write a program to convert the given temperature in Fahrenheit to Celsius using the following conversion formula  $C = (F - 32)/1.8$  And display the values in a tabular form. [8+8]
2. (a) What is the purpose of using a method? How do you declare a method? How do you invoke a method?  
(b) What is method overloading? Can you define two methods that have same name but different parameter types? Can you define two methods in a class that have identical method names and parameter profile with different return value types or different modifier ? [8+8]
3. Add a new method in the base class of Shapes.java that prints a message, but don't override it in the derived classes. Explain what happens. Now override it in one of the derived classes but not the others, and Explain what happens. Finally, override it in all the derived classes, Explain in detail about each situation. [16]
4. (a) What is a package? How do we design a package?  
(b) How do we add a class or interface to a package? [8+8]
5. (a) Define multithreading. Give an example of an application that needs multithreading.  
(b) How multithreading in single processor system is different from multithreading in multiprocessor system. Explain. [6+10]
6. Write a java program which creates human face. [16]
7. What are various JFC containers? List them according to their functionality. Explain each of them with examples. [16]

8. (a) Discuss briefly about the following: TCP, UDP, URL  
(b) What is InetAddress? How to create an InetAddress? What is its use? [8+8]

**Code No: R05220402 Set No. 3**

**II B.Tech II Semester Regular Examinations, Apr/May 2008**

**OOP THROUGH JAVA**

( Common to Electronics & Communication Engineering, Computer Science & Engineering, Information Technology, Computer Science & Systems Engineering, Electronics & Telematics and Electronics & Computer Engineering)

Time: 3 hours Max Marks: 80

Answer any FIVE Questions

All Questions carry equal marks

1. (a) Describe the genesis of java. Also write brief overview of java.  
(b) Write a program to convert the given temperature in Fahrenheit to Celsius using the following conversion formula  $C = (F - 32)/1.8$  And display the values in a tabular form. [8+8]
2. (a) What is an array? Why arrays are easier to use compared to a bunch of related variables?  
(b) Write a program for transposition of a matrix using arraycopy command. [6+10]
3. Create a base class with an abstract print( ) method that is overridden in a derived class. The overridden version of the method prints the value of an int variable defined in the derived class. At the point of definition of this variable, give it a nonzero value. In the base-class constructor, call this method. In main( ), create an object of the derived type, and then call its print( ) method. Explain the results. [16]
4. Write a program create an interface U with three methods. Create a class A with a method that produces a reference to a U by building an anonymous inner class. Create a second class B that contains an array of U. B should have one method that accepts and stores a reference to a U in the array, a second method that sets a reference in the array (specified by the method argument) to null and a third method that moves through the array and calls the methods in U. In main( ), create a group of A objects and a single B. Fill the B with U references produced

by the A objects. Use the B to call back into all the A objects. Remove some of the U references from the B. [16]

5. (a) Explain throws statement in Java with the help of an example program.

(b) What is the difference between throw and throws statement. [8+8]

6. (a) Write a java program which draws a dashed line and dotted line using applet.

(b) Write a java program to draw a polygon of eight edges. [10+6]

7. (a) In what way JList differ from JComboBox?

(b) JList does not support scrolling. Why? How this can be remedied? Explain with an example. [6+10]

8. Define sockets. Use socket programming to design a client/server application that takes the password as input and checks whether it is correct. The program should print the appropriate message. [16]

**Code No: R05220402 Set No. 4**

**II B.Tech II Semester Regular Examinations, Apr/May 2008**

**OOP THROUGH JAVA**

( Common to Electronics & Communication Engineering, Computer Science & Engineering, Information Technology, Computer Science & Systems Engineering, Electronics & Telematics and Electronics & Computer Engineering)

Time: 3 hours Max Marks: 80

Answer any FIVE Questions

All Questions carry equal marks

1. Write a program that will compute the following series:

(a)  $1/1 + 1/2 + 1/3 + \dots + 1/n$

(b)  $1/1 + 1/2 + 1/2$

$2$

$+ \dots + 1/2^n$ . [8+8]

2. (a) What is a constructor? What are its special properties?

(b) How do we invoke a constructor?

(c) What are objects? How are they created from a class? [6+4+6]

3. Explain about Object class in detail. [16]

4. (a) What is interface? Write a program to demonstrate how interfaces can be

extended.

(b) What is package? How do you create a package? Explain about the access protection in packages? [8+8]

5. (a) Explain how threads with different priorities execute in environment which supports priorities and which doesn't support priorities.

(b) what are the functions available in java related to priority. [10+6]

6. (a) Why do you use frames?

(b) Explain the syntax and functionality of different methods related to Frames.

[4+12]

7. Explain the steps involved in creating JCheckBox, JRadioButton, JButton, JLabel.

[4+4+4+4]

8. Write a program to illustrate the usage of the following methods of StringBuffer class. Explain the output in each case. Delete(), setCharAt(), deleteCharAt(), append(), charAt(), getChars(). [16]

## 17 Question Bank

### QUESTION BANK

#### UNIT I

1.
  - a) What Kind things can become objects in OOP?
  - b) List few areas of application of OOP technology.
2.
  - a) Explain different types of class hierarchies.
3.
  - a) What makes java a robust language? Explain.
  - b) Explain about the class hierarchies.
4.
  - a) Explain about the method overriding in Java.
  - b) Write a java program to implement the method overriding.
5.
  - a) Explain the way of viewing the world agents to oops.
  - b) Explain the way of viewing the world responsibilities about oops.
6.
  - a) What are the advantages of an object – oriented programming paradigm.
  - b) Explain the fundamental characteristics of OOPS.
  - c) Describe with a flow chart how various tools are used in the application development.
7.
  - a) Describe the advantages of object oriented technology.
  - b) How is Java strongly associated with the intern

8.
  - a) Explain the need for object oriented programming paradigm.
  - b) What are the components of Java Architecture? Explain in detail?
  - c) Describe with a flow chart how various tools are used in application development.
9.
  - a) Explain about Java class libraries.
  - b) Explain about relational and logical operators with examples.
10.
  - a) Describe the genesis of java. Also write brief overview of java.
  - b) Write a program to convert the given temperature in Fahrenheit to Celsius using the following conversion formula  $C = (F - 32)/1.8$  And display the values in a tabular form.
11. Explain briefly the main concepts of object-oriented programming.
12.
  - a) How are data and methods organized in an object-oriented program?
  - b) What are unique advantages of an object-oriented programming paradigm?
13. What are the chief weakness of procedural languages? How are they covered in object oriented programming languages?
14. What are the most important common features of procedural languages and object oriented programming languages?
15. Write a brief notes on the following:
  - a) Data hiding
  - b) Classes and objects
  - c) Encapsulation
  - c) Polymorphism
16. What is object oriented programming? How is it different from procedure-oriented programming?
17. Contrast: Object-based Vs Object-oriented. Give an example (other than Java and C++) of each category of languages
18. How data and methods organized in an object oriented program?
19. Write a program for transposition of a matrix using array copy command.
20. Explain the following methods of the String Buffer class. length ( )
  - o Capacity ( )
  - o Ensure Capacity ( )
  - o Set Length ( ) .
21. Write an application that computes the value of ex by using the formula:  
 $ex = 1 + x/1! + x^2/2! + x^3/3! + \dots (4+4+8)$

22. Write a program that will compute the following series:
- $1/1 + 1/2 + 1/3 + \dots + 1/n$
  - $1/1 + 1/2 + 1/2 + \dots + 1/2n$ . (8+8)
23. List and explain the control statements used in java. Also describe the syntax of the control statements with suitable illustration.
24. What is the difference between a public member and a private member of a class? (Nov 06/Feb 0708)
25. What are objects? How are they created from a class?
26. How the constructor in Java is implemented? What are its special properties? How many types of constructors are there in Java? Explain.
27. What are constructors and destructor functions? Explain different types of constructors?
28. How do we invoke a constructor?
29. What is method overloading? Can you define two methods that have same name but different parameter types? Can you define two methods in a class that have identical method names and parameter profile with different return value types or different modifier?
30. Write a Java program that illustrates the usage of the charAt() and setCharAt () methods of the String Buffer class. Give the output.
31. What are variables? Discuss about the scope and life time of variables in detail.
32. a) Explain about 'this' keyword with an example.  
b) Differentiate between C++ and java?  
c) Write a program to illustrate specification and explain it.
33. a) What is recursion? Explain in detail.  
b) Write a program to display Fibonacci series.
34. a) Briefly explain Stream Tokenizer class.  
b) Write a program to illustrate the use of stream tokenizer class.
35. a) Explain about type conversion with an example.  
b) Explain about wrapper class with an examples.
36. a) Write a Java Program to find the greatest common divisor of two numbers.  
b) Compare and contrast between while and do – while statement.  
c) Explain the constructor method. How it differs from other member function?
37. a) Compare and contrast between concrete class and abstract class.  
b) Write a program to check whether given string is palindrome or not.  
c) Write a program to find whether the given triangle is right angled or not
38. a) Describe with a flow chart how various tools are used in the applet development.

- b) Java is called the “Platform independent language”. Why is it Platform independent?
  - c) What is a class? How do classes help us to organize our programs?
39. a) Differentiate between static and instance methods.
- b) Write a program to check the given number is prime or not.
  - c) Write short notes on
    - i) this
    - ii) transient
    - iii) volatile
40. a) Write a program to solve quadratic equations.
- b) What is type casting? What are the rules followed for type casting?
  - c) Explain the following JAVA key words : (i) static (ii) final
41. Write a program to display multiple lines of text in a window. It also displays multiple sentences on the same line.
42. a) Write a Java Program for Simple chatting using TCP.
- b) Explain about the Factory methods.
43. a) What is multiple inheritance? Explain how does Java support multiple inheritance?
- b) Write about the data types supported by Java?
44. a) How does String class differ from the StringBuffer class?
- b) Write a program to extract a portion of a character string and print the extracted string .assume that m characters are extracted, starting with the nth character.
  - c) Write a program ,which will read a text and count all occurrences of a particular word.
- List at least ten major differences between C and Java?
- 45 Write a program to convert the given temperature in Fahrenheit to Celsius using the following conversion formula  $C = (F - 32)/1.8$  and display the values in a tabular form.
46. What is an empty statement? Explain its usefulness.
47. Explain the difference between an object and a class.
48. Write a program in order to find out what the following statement will do for ( ; ) some thing.
49. How a for loop can be defined and used within the program? Explain with suitable examples.
50. How java is associated with the Internet?
51. Can a java run on any machine? What is needed to run java on a computer?
52. List the modifiers and describe their purposes.
53. How do you pass actual parameters to a method? Can actual parameter have the same name as its formal parameters?
54. Explain how to call virtual functions in base class constructor?

55. Explain usage of Command line arguments with an example.
56. How can a base class protect derived classes so that changes to the base class will not affect them? Explain? (Feb 07)
57. When should a data member be protected: rather than private?
58. Why is private: the default access level for a class?
59. When do you declare a member function of a class static?
60. Write a while loop to find the smallest n such that n<sup>2</sup> is greater than 10,000.
61. What is the output of the following program code?  
int m=100;  
int n=300;  
While (++m< n), System.out.println(m);  
(Feb 2007)
62. In what ways does a switch statement differ from an if statement. Give examples.
63. Write a program to find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7.

## UNIT II

1. What is inheritance? Explain the member access mechanism in inheritance with an example.
2. Create an abstract class with no methods. Derive a class and add a method. Create a static method that takes a reference to the base class, downcasts it to the derived class, and calls the method. In main ( ), demonstrate that it works. Now put the abstract declaration for the method in the base class, thus eliminating the need for the downcast.
3. What is Inheritance? Discuss the differences in inheritances in C++ and java.
4. Discuss different types of Inheritances in Interfaces.
5. a) Define multiple inheritance. Does Java Support multiple inheritance. Justify your answer.  
b) Write java program to implement the multilevel Inheritance.
6. a) How do you construct a class from another class? Explain with an example.  
b) Can a sub class access all the members of super class? Justify your answer.
7. a) What is the use of “this” Keyword explain with an example.  
b) Write a program that creates an abstract class called dimension, creates two subclasses, rectangle and triangle. Include appropriate methods for both the subclasses that calculate and display the area of the rectangle and triangle.
8. a) Describe the different levels of access protections available in Java with an example. (May 10)

- b) Write a program to illustrate the method overriding.
- 9. a) Differentiate between composition and inheritance.
- b) Write a method find\_Area that can find the area of the circle/square/rectangle using overloading concept.
- c) What is the use of Super Keyword?
- 10. a) Differentiate between method overloading and method overriding.
- b) Write a program to illustrate the dynamic method dispatch.
- 11. a) Write a Java program that uses the length() and capacity() methods of the String Buffer class.

Give the output.

- b) Explain the charAt() and setCharAt() methods of the StringBuffer class.
- 12. a) Discuss about Hybrid Inheritance with a suitable example.
- 13. Discuss about Hierarchical Inheritance with a suitable example
- 14. Explain about final classes, final methods and final variables?
- 15. Explain about the abstract class with example program?
- 16. Justify the following statement with an example. “A super class variable can reference a subclass object”.
- a) Explain the main two uses of super.
- b) Explain the procedure to call super class members with example.
- 17. Create a 3-level inheritance hierarchy. Each class in the hierarchy should have a finalize( ) method, and it should properly call the base-class version of finalize( ). Demonstrate that your hierarchy works properly. (Nov 07)
- 18. Is there any alternative solution for Inheritance? If so explain the advantages and disadvantages of it.
- 19. a) Define Abstract class? Explain with a suitable example.
- b) Write a sample program to demonstrate the order of initialization of the base classes and derived classes. Now add member objects to both the base and derived classes, and show the order in which their initialization occurs during construction.
- 20. Discuss the different ways by which we can access public member functions of an object?
- 21. Create a base class with an abstract print( ) method that is overridden in a derived class. The overridden version of the method prints the value of an int variable defined in the derived class. At the point of definition of this variable, give it a non zero value. In the base-class constructor, call this method. In main( ), create an object of the derived type, and then call its print( ) method. Explain the results.

22. Write about abstract class and early binding.
23. Compare and contrast overloading and overriding methods.
24. Explain about Object class in detail.
25. a) What is inheritance and how does it help us to create new classes quickly?  
b) Describe the different forms of inheritance with examples?
26. What are the types of inheritances in java? Explain each of them in detail.
27. a) What are the differences between private, final and static variables?  
b) Explain about Dynamic Method Dispatch.
28. Write a Super class interface employee has name and id number. Write manager and labour derived from employee class. Manager class has member data qfunction and qualification and manager allowance & rank Labour class has member data Dailywage, Overtime & grade.
29. When do we declare a method or class abstract?
30. How can you prevent a class from instantiation?
31. When do we declare a method or class final?
32. Describe different levels of access protections available in java.
33. Describe the various forms of implementing inheritances? Give an example of code for each case?
34. a) Explain in detail about accessing a package.  
b) Write short note on CLASSPATH environmental variable.
35. Create an Interface called Integer stack. Write a program that uses this interface to implement fixed stack and dynamic stack.
36. a) What is the difference between implicit and explicit import statement? Which one take less time for compilation?  
b) How to extend one interface by the other interface? Explain with an example.
37. a) What is the usage of javac -d option?  
b) How to design and implement an interface. Explain with example?  
c) Explain implicit and explicit import statement.
38. a) Define a package? What is the necessity of packages?  
b) Explain implicit and explicit import statement.  
c) What is the major difference between an interface and class?
39. a) Compare and contrast between class and an interface.  
b) Write a program to implement a class Teacher contains two fields Name and Qualification. Extend the class to Department, it contains Dept. No and Dept. Name. An Interface named as

College it contains one field Name of the College. Using the above classes and Interface get the appropriate information and display it.

40. Prove that the fields in an interface are implicitly static and final.
41. What is package? Explain the procedure to create a package with the help of example.
42. Write a program to create a class with a non default constructor and no default constructor. Create a second class that has a method which returns a reference to the first class. Create the object to return by making an anonymous inner class that inherits from the first class.
43. Write a program to create a private inner class that implements a public interface. Write a method that returns a reference to an instance of the private inner class, upcast to the interface. Show that the inner class is completely hidden by trying to downcast to it.
44. Write a program create an interface U with three methods. Create a class A with a method that produces a reference to a U by building an anonymous inner class. Create a second class B that contains an array of U. B should have one method that accepts and stores a reference to a U in the array, a second method that sets a reference in the array (specified by the method argument) to null and a third method that moves through the array and calls the methods in U. In main( ), create a group of A objects and a single B. Fill the B with U references produced by the A objects. Use the B to call back into all the A objects. Remove some of the U references from the B.
45. Create an interface with at least one method, in its own package. Create a class in a separate package. Add a protected inner class that implements the interface. In a third package, inherit from your class and, inside a method, return an object of the protected inner class, up casting to the interface during the return.
46. Prove that all the methods in an interface are automatically public.
47. Write a sample program to illustrate packages
48. a) What is a package? How do we design a package?  
b) How do we add a class or interface to a package?
49. a) Give general form of the package statement. Give an example package creation statement.  
b) Give general form of a multileveled package statement. What is the significance of the CLASSPATH environment variable in creating/using a package  
c) Give the general form of the import statement. Illustrate a Java program that creates a package and uses it.

50. Write a program to create an interface containing a static inner class. Implement this interface and create an instance of the inner class.
51. Write an interface called shape with necessary methods. Derive classes circle, rectangle, triangle, cone, sphere and cube with appropriate constructors and methods for area, volume also setting & displaying.
52. a) What is interface? Write a program to demonstrate how interfaces can be extended.  
b) What is package? How do you create a package? Explain about the access protection in packages?
53. a) Define an interface? Explain the difference between class and interface. List out the various interfaces in java.  
b) Write a program to get n numbers from the users and print and largest numbers.
54. Write an interface called shape with necessary methods. Derive classes circle, rectangle, triangle, cone, sphere and cube with appropriate constructors and methods for area, volume also setting & displaying.

### UNIT III

1. What is the role of stack in exception handling?
  - a) Give the classification of exceptions.
  - B) Give the list of different checked exceptions in java and their meaning.
2. a) What is meant by uncaught exceptions? Discuss how to deal with them. Explain it with a sample Java program.  
b) Discuss the key terms \throw" and \throws". Give suitable examples which shows how to use them.
3. What are the Java's built-in exceptions? List the checked exceptions defined in the Java:lang and explain them clearly with suitable examples.
4. a) Discuss about multiple catch clauses and nested try statements.  
b) Explain the following:
  - i. ArithmeticException
  - ii. ArrayStoreException
  - iii. ClassCastException
  - iv. IllegalStateException
5. List out various classes in \Java.util" and explain them clearly.
6. List out the classes in Java.util Package along with their purpose.

7. a) What is Synchronization? Explain with suitable example.  
b) Write a program that generates a user defined Exception.
8. a) Explain how exception handling mechanism can be used in a program.  
b) Write a Java Program to implement Runnable class to create a thread.
9. a) What is an exception? How can java handle the exceptions? Illustrate with an example.  
b) Explain about various key words used in handling the exceptions.
10. Explain the following Thread related exceptions with examples:
  - a) IllegalMonitorStateException
  - b) IllegalThreadStateException.
11. Explain in detail any three checked exceptions.
12. Explain the following exceptions with the help of examples:  
ArithmeticException  
NullPointerException
13. If we try to catch a super class exception type before a sub class type, compiler generates exception errors. Explain why this error occurs with an example?
14. What is the difference between unchecked and checked exceptions in java?
15. Give the list of different unchecked exceptions in java and their meaning.
17. Explain in detail any two unchecked exceptions.
18. a) Explain how threads with different priorities execute in environment which supports priorities and which doesn't support priorities.  
b) What are the functions available in java related to priority?
19. What is an exception? How can java handle the exceptions? Illustrate with an example. Explain about various key words used in handling the exceptions.
20. Why exception handling is considered as one of the important features in OOPS? Write your explanation with suitable examples.
21. Write a java program that illustrates re-throwing an exception.
  - a. What is the use of five keywords of Java related to exception handling i.e., try, catch, throw, throws, and finally.
  - b. Write an example Java program using all the five key words mentioned above, and explain how the program works.
22. Explain in detail any three checked exceptions.

23. A program throws an exception and the appropriate exception handler begins execution, and this exception handler in turns throws the same exception. Is this above approach creating an infinite recursion? Justify your answer with an example
24. What is meant by multithreaded programming? Discuss about Java thread model, thread priorities and interthread communication.
25. a) Discuss about wait( ), notify( ) and notifyAll( ) methods in Java.  
b) What is meant by a Deadlock? Is it possible to occur in a multithreaded program? Justify your answer with a sample Java program.
26. a) Discuss various methods of object class.  
b) Write a program that illustrates notify, notifyAll( ) and wait( ) methods of object class.
27. a) Explain clearly the Single-member Annotations with a sample Java program.  
b) Discuss the Autoboxing/Unboxing Boolean and character values.
28. Discuss in detail about various Enumerations and Annotations in Java. Write sample Java programs to describe each of them.
29. a) Write a program that demonstrate the priority setting in threads.  
b) Write an example program for multithreading using thread class.
30. a) How do we set priorities for threads? Describe the complete life cycle of a thread.  
b) Write a program that includes a try block and a catch clause which processes the arithmetic exception generated by division – by – zero error.

#### **UNIT-IV**

1. Write about various Stream Classes in java.
2. Discuss about the File Input stream and File Output Stream in java with examples
3. Write short notes on java.io package and java.lang package.
4. Write about the various Character Streams in java.
5. Write about the various Byte Streams in java?
6. Explain the process of connecting to a database
7. Write short notes on types of JDBC drivers
8. explain file handling using File class
9. Write short notes on collections frame work of java
10. write a program illustrating following collections framework
  - a) ArrayList
  - b) Vector
  - c) HashTable
  - d) Stack

## UNIT-V

1. Write a stand-alone AWT based application which) creates a frame window that responds to mouse clicks and key strokes.
2.
  - a) What is the functionality supported by java related to drawing ellipses and circles.
  - b) What is the functionality supported by java related to drawing arcs.
3.
  - a) How do you scale a drawing object in java? Explain with an example java program.
  - b) What is the functionality supported by java related to colours.
4. What are the components and other graphical user interface elements that can act as source of events? What are the events that can be generated by the above elements? Explain.
5. What are the methods supported by the following interfaces. Explain each of them
  - a) ActionListener interface
  - b) MouseMotionListener interface
  - c) TextListener interface.
6. What are the methods supported by KeyListener interface and MouseListener interface. Explain each of them with examples.
7. With the help of an example program explain how you handle all mouse related events.
8.
  - a) What is the functionality supported by java related to Fonts.
  - b) How using different fonts improves the user interface.
9. Explain different event classes supported by Java.
10.
  - a) Distinguish between Grid layout and Grid bag layout.
  - b) Give the differences between panel and frame.
  - c) What is an adapter class? Describe about the Mouse adapter class.
11. Explain the different types of Layout managers with suitable examples.
12.
  - a) Distinguish between Grid layout and Grid bag layout.
  - b) Give the differences between panel and frame.
  - c) What is an adapter class? Describe about the Mouse adapter class.
13. Write a program to illustrate Grid bag layout.
14. Differentiate following with sample programs.
  - a) Grid & Grid bag layout
  - b) Card & border layout.
15.
  - a) How event driven programming is different from Procedure oriented programming.
  - b) Give overview of Java's event handling mechanism.
16. Explain in detail about the following event classes:

- a) `ComponentEvent`
- b) `ContainerEvent`
- c) `FocusEvent`

17. Explain Grid layout Manager with an example.
18. a) Write Java code for an Applet skeleton and explain it.  
b) What are the methods used for Applet initialization and termination? Explain them clearly.
19. a) Write the HTML Applet Tag and explain each part of it.  
b) What are the four forms of the method `repaint ( )`? Explain their usage with a sample Java program.
15. a) Differentiate between an Applet and an Application.  
b) Write a Java Program to display the following 3 by 3 Magic Square (Total = 15) using `JTable`.

2	9	4
7	5	3
6	1	8

16. What is the use of `JFrame`? Create a `JFrame` containing a `JDesktoppane`, which has a single `internalFrame`.
17. What are various JFC containers? List them according to their functionality.
18. a) Discuss various integer constants, constructors and methods defined in the `KeyEvent` class.  
b) Explain the `FocusEvent` class and the `InputEvent` class.
19. Discuss the following Event Listener Interfaces and also discuss various methods declared in it.
  - a) `WindowFocusListener`
  - b) `TextListener`
  - c) `KeyListener`
  - d) `MouseListener`
20. a) Explain the `AdjustmentEvent` class and its usage with a sample program.  
b) Discuss about various sources of events.
21. Write a Java Program to illustrate menu.
22. Write a program to create frame window that responds to mouse clicks and keystrokes.
23. Write a program to handle HTTP get request?
24. Differentiate choice lists and scrolling lists? How will you add them to an applet? Explain with suitable examples.
25. Create an applet with two toolbars. One toolbar should be created using `JButtons` and a separator and another toolbar should be created using 3 custom Action classes. Add one to the "north" and

another to the "south" sides of border layout. When the user clicks one of the buttons in the toolbar, it will print a message to the console stating that which button is being pressed from which toolbar. Add functionalities to the buttons such as New, Open, Close, Save, Cut, Copy, Paste.

26. Explain the steps involved in creating a frame. Create a desktop pane, add it to a frame and display two internal frames in it. Internal frames should have 2 different layers.
27. a) How will you create a menu in java?  
b) Write a frame application that has an edit menu, which consists of cut, copy and paste and performs the same functions when selected. Create a text box to operate on.
28. a) Explain menu class and menu actions.  
b) Is it possible to create nested menus and submenus? If so, how?
29. Explain various text components in JFC with examples.
30. a) Why creating a subclass of Frame is preferred over creating an instance of Frame when creating a window.  
b) Explain the steps in creating a subclass of frame with the help of examples.
31. a) What is the use of JPasswordField? Explain with an aid of an application program.  
b) What are the differences between JPopupMenu and JMenu?
32. Differentiate following with suitable examples:
  - a) Frame, JFrame
  - b) Applet, Japplet L53
  - c) Menu, Jmenu.
33. a) Explain various components of User Interface.  
b) How will you arrange components on User Interface?
34. Write a program that randomly draws characters in different font sizes and colors.
35. Write a program that randomly draws characters in different font sizes and colors.
36. What are the methods supported by Key Listener interface and Mouse Listener in-terrace. Explain each of them with examples.
37. a) Explain menu class and menu actions.  
b) Is it possible to create nested menus and submenus? If so, how?
38. a) Why do you use frames?  
b) Explain the syntax and functionality of different methods related to Frames.
39. What is meant by AWT? How will you create User Interfaces for applets?

40. a) What is the use of JPasswordField? Explain with an aid of an application program.  
b) What are the differences between JPopupMenu and JMenu?
41. Explain the functionality of JComponent with example. Differentiate JComponent and JPanel.
42. a) Explain the goals of Swing.  
b) Explain the Swing architecture with a diagram.
43. Discuss about various methods defined in the following classes:
- |               |            |
|---------------|------------|
| a) ImageIcon  | b) JLabel  |
| c) JTextField | d) JButton |

## 18 Assignment Questions

### Assignment-1

1. Differentiate between procedural and object oriented languages
2. Write a program that finds the largest prime number that is less than a specified value.
3. Write a Java Program to represent bank account containing Account number account name and balance and methods as Deposit and withdraw.
4. Write a Java Program to Sort a given list of names in ascending order.
5. Write a Java Program to check whether a String is palindrome or not.
6. Write a Java Program to Count number of words in a given text.
7. Write a Java Program to demonstrate Narrowing conversion.
8. Write a Java Program to demonstrate Constructor overloading.
9. Write a Java Program to demonstrate Constructor chaining.
10. Write a Java Program to demonstrate String buffer class methods.

### Assignment-2

1. Write a Java Program to demonstrate Method overriding.
2. Write a Java Program demonstrate the Use of “super” keyword.

3. Write a Java Program to create Geometric shape interface with methods area( ) and perimeter( ) for classes triangle ,rectangle and circle.
4. Write a Java Program to demonstrate Packages containing package name as “sortapp” a in which declare an interface “sortInterface” with method sort() whose return type and parameter type should be void and empty define “subsortapp “ as subpackage of “sortapp” package in which define a class.
5. Write a Java Program to Sort a given list of names in ascending order.
6. Write a Java Program to represent bank account containing Account number account name and balance and methods as Deposit and withdraw.
7. Write a Java Program to demonstrate Single inheritance.
8. Write a Java Program to demonstrate Static keyword.
9. Design and develop a java program to show abstract methods

### Assignment-3

1. Write a Java Program Method to throw “insufficient fund exception” in withdraw method of bank account class.
2. Write a Java Program to define two threads such that one thread prints even numbers and another thread prints odd numbers.
3. Write a Java Program to demonstrate Synchronization concept.
4. Design and develop a java program to create three threads the first thread prints ‘good morning’ for every one second ,the second thread prints ‘hello!’ for every 2 seconds and the third thread prints ‘welcome’ for every three seconds.
5. Write a Java Program to demonstrate Event handling.
6. Write a Java Program to demonstrate Try-catch-finally all in one program.
7. Design and develop a java program for creating our own Exception class. In this the details of account number, customer name and balance amount in the form of 3arrays the in main() method we display these details using for loop. at this time we check if any account balance is less than 1000/- the MyException is raised and a message is displayed like ‘ balance is less than 1000/-“.

#### **Assignment-4**

1. Design and develop a java program that reads a filename and displays the file on screen with the line number before each line
2. Design and develop a java program that reads a filename from keyboard and display the number of characters, lines, and words in the file.
3. Design and develop a java program that reads a filename from the user that displays information about whether the file exists\readable\writable\type of file and length of file.
4. Design and develop a java program to show all the stack operations.
- 5 Illustrate Vector and Hashtable using a program.

#### **Assignment-5**

1. Write a Java Program to demonstrate Choice.
2. Write a Java Program to demonstrate Menu Bar.
3. Design and develop a java program to create an applet and display a simple message like “Hello! Welcome to Applet Programming”.
4. Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time No light is on when the program starts.
5. Write a Java program that allows the user to draw lines, rectangles and ovals.
6. Design and develop a java program to show the use Calendar class
7. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -,\*, % operations. Add a text field to display the result.
8. Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the textfields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the

program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException Display the exception in a message dialog box.

## 19 Unit wise Quiz Questions and long answer questions

### UNIT-1

1. \_\_\_\_\_ is the ability for a message or data to be processed in more than one form.
  - a. inheritance
  - b. abstraction
  - c. encapsulation
  - d. polymorphism
2. An \_\_\_\_\_ is a software bundle of variables and related methods
3. Which of the following statement about the java language is true?
  - a) Both procedural and OOP are supported in java
  - b) Java supports only procedural approach through programming
  - c) Java supports only OOP approach
  - d) None of the above
4. Which of the following statements is false about objects?
  - a) An instance of a class is an object
  - b) Objects can access both static and instance data
  - c) Object is the superclass of all other classes
  - d) Object do not permit encapsulation
5. \_\_\_\_\_ is a technical definition of the language that includes the syntax and semantics.
6. \_\_\_\_\_ can be characterized as data controlling access to code.
7. \_\_\_\_\_ is a software unit that combines a structured set of data with a set of operations for inspecting and manipulating that data.
8. \_\_\_\_\_ is a group of instructions that is given a name and can be called up at any point in a program simply by quoting that name
  - a. method
  - b. class
  - c. data
  - d. messages
9. \_\_\_\_\_ allows to reuse classes by deriving a new class from an existing one
  - a. inheritance
  - c. encapsulation

b. abstraction                      d.polymorphism

10. \_\_\_\_\_ is an object that can be used to call an arbitrary public method, on an instance that is acquired by evaluating the leading portion of a method binding expression via a value binding

a.Method binding   b. Type binding   c. a&b   d.none

11. \_\_\_\_\_ is an interpreter for bytecode.

b.

a. Compiler

JVM

c. Internet

d. ACM

12.The class that inherits is called a

a.Superclass

b.subclass

c.virtual

d.instance class

13.The \_\_\_\_\_ operator is used to create a new object or a new array.

a.instanceOf

b.scope

c.(.)

d.new

14.A constructor is used to

a.destroy

memory

b.initialize newly created object

c.import packages

d.create a JVM for applets

15.In java, the equal sign is used as the

a.increment

b.decrement

c.assignment

d.negation

16.To terminate a program, use the java

statement:

a.System.quit(0);

d.System.exit(0);

b.System.end(0);

c.System.abort(0);

17.Which method can access to private attributes of a class?

a)Only static method of the same class

b)Only instances of the same class

c)Only methods those defined in the same class

d)Only classes available in the same package

18.What is the value of 'number' after the following code fragment execution?

```
int number = 0;
int number2 = 12
while (number < number2)
{
number = number +1;
}
```

a)5            b)12            c)21            d)13

19. Which can be passed as an argument to a function?

a)Constant            b)Expression            c)Other function            d)All the above

20. "this" pointer

a)Implicitly points to an object            b)Can be explicitly used in class  
c)Can be used to return an object            d)All the above

21. Static function Should be called when an object is destroyed

- a. Is closely connected with individual objects of a class
- b. Can be called using the class name and function name
- c. Is used when a dummy object must be created
- d. none

22. The new operator [ ]

- a) Returns a pointer to the variable            b) Creates a variable called new
- c) Obtains memory for a new variable            d) Tells how much memory is available

23. \_\_\_\_\_ is the region where the code that needs to be executed under circumstances is written.

24. \_\_\_\_\_ is an independent path of execution in a program.

25. A program that a java enabled browser can download and run is

a. \_\_\_\_\_

26. What is the range of data type short in Java?  
a) -128 to 127 b) -32768 to 32767 c) -2147483648 to 2147483647 d) None of the mentioned

27. What is the range of data type byte in Java?  
a) -128 to 127 b) -32768 to 32767 c) -2147483648 to 2147483647 d) None of the mentioned

28. Which of the following are legal lines of Java code?

1. `int w = (int)888.8;`
  2. `byte x = (byte)100L;`
  3. `long y = (byte)100;`
  4. `byte z = (byte)100L;`
- a) 1 and 2 b) 2 and 3 c) 3 and 4 d) All statements are correct.

29. An expression involving byte, int, and literal numbers is promoted to which of these?

- a) int b) long c) byte d) float

30. Which of these literals can be contained in a data type float variable?

- a) 1.7e-308 b) 3.4e-038 c) 1.7e+308 d) 3.4e-050

31. Which data type value is returned by all transcendental math functions?

- a) int b) float c) double d) long

32. What is the output of this program?

```
1.      class average {  
2.          public static void main(String args[])  
3.          {  
4.              double num[] = {5.5, 10.1, 11, 12.8, 56.9, 2.5};  
5.              double result;  
6.              result = 0;  
7.              for (int i = 0; i < 6; ++i)  
8.                  result = result + num[i];  
9.                  System.out.print(result/6);  
10.     }
```

11.            }

12.            }

a) 16.34 b) 16.566666644 c) 16.466666666666667 d) 16.466666666666666

33. What is the output of this program?

1.            **class** conversion {

2.            **public static** void main(String args[])

3.            {

4.            double a = 295.04;

5.            int b = 300;

6.            byte c = (byte) a;

7.            byte d = (byte) b;

8.            System.out.println(c + " " + d);

9.            }

10.           }

a) 38 43 b) 39 44 c) 295 300 d) 295.04 300

34. What is the output of this program?

1.            **class** increment {

2.            **public static** void main(String args[])

3.            {

4.            int g = 3;

5.            System.out.print(++g \* 8);

6.            }

7.            }

a) 25 b) 24 c) 32 d) 33

35. What is the output of this program?

1.            **class** area {

2.            **public static** void main(String args[])

3.            {

```
4.         double r, pi, a;
5.         r = 9.8;
6.         pi = 3.14;
7.         a = pi * r * r;
8.         System.out.println(a);
9.     }
10.    }
```

a) 301.5656 b) 301 c) 301.56 d) 301.56560000

36. What is the numerical range of a char in Java?

a) -128 to 127 b) 0 to 256 c) 0 to 32767 d) 0 to 65535

37. Which of these coding types is used for data type characters in Java?

a) ASCII b) ISO-LATIN-1 c) UNICODE d) None of the mentioned

38. Which of these values can a boolean variable contain?

a) True & False b) 0 & 1 c) Any integer value d) true

39. Which of these occupy first 0 to 127 in Unicode character set used for characters in Java?

a) ASCII b) ISO-LATIN-1 c) None of the mentioned d) ASCII and ISO-LATIN1

40. Which one is a valid declaration of a boolean?

a) boolean b1 = 1; b) boolean b2 = 'false'; c) boolean b3 = false; d) boolean b4 = 'true'

41. What is the output of this program?

```
1.     class array_output {
2.         public static void main(String args[])
3.         {
4.             char array_variable [] = new char[10];
5.             for (int i = 0; i < 10; ++i) {
6.                 array_variable[i] = 'i';
7.                 System.out.print(array_variable[i] + " ");
8.                 i++;
9.             }
```

10.            }

11.            }

a) i i i i I b) 0 1 2 3 4 c) i j k l m d) None of the mentioned

42. What is the output of this program?

```
1.            class mainclass {
2.                public static void main(String args[])
3.                {
4.                    char a = 'A';
5.                    a++;
6.                        System.out.print((int)a);
7.                }
8.            }
```

a) 66 b) 67 c) 65 d) 64

43. What is the output of this program?

```
1.            class mainclass {
2.                public static void main(String args[])
3.                {
4.                    boolean var1 = true;
5.                    boolean var2 = false;
6.                    if (var1)
7.                        System.out.println(var1);
8.                    else
9.                        System.out.println(var2);
10.                }
11.            }
```

a) 0 b) 1 c) true d) false

44. What is the output of this program?

```
1.            class booloperators {
```

```
2.      public static void main(String args[])
3.      {
4.          boolean var1 = true;
5.          boolean var2 = false;
6.          System.out.println((var2 & var2));
7.      }
8.      }
```

a) 0 b) 1 c) true d) false

45. What is the output of this program?

```
1.      class asciicodes {
2.          public static void main(String args[])
3.          {
4.              char var1 = 'A';
5.              char var2 = 'a';
6.              System.out.println((int)var1 + " " + (int)var2);
7.          }
8.      }
```

a) 162 b) 65 97 c) 67 95 d) 66 98

46. Which of these is data type long literal?

a) 0x99ffL b) ABCDEFG c) 0x99ffa d) 99671246

47. Which of these is returned by operators &, ?

a) Integer b) Boolean c) Character d) Float

48. Literals in java must be preceded by which of these?

a) L b) l c) D d) L and I

49. Literal can be of which of these data types?

a) integer b) float c) Boolean d) all of the mentioned

50. Which of these can not be used for a variable name in Java?

a) identifier b) keyword c) identifier & keyword d) None of the mentioned

51. What is the output of this program?

```
1.      class evaluate {
2.          public static void main(String args[])
3.          {
4.              int a[] = {1,2,3,4,5};
5.              int d[] = a;
6.              int sum = 0;
7.              for (int j = 0; j < 3; ++j)
8.                  sum += (a[j] * d[j + 1]) + (a[j + 1] * d[j]);
9.              System.out.println(sum);
10.         }
11.     }
```

a) 38 b) 39 c) 40 d) 41

52. What is the output of this program?

```
1.      class array_output {
2.          public static void main(String args[])
3.          {
4.              int array_variable [] = new int[10];
5.              for (int i = 0; i < 10; ++i) {
6.                  array_variable[i] = i/2;
7.                  array_variable[i]++;
8.                  System.out.print(array_variable[i] + " ");
9.                  i++;
10.             }
11.
12.     }
```

13.            }

a) 0 2 4 6 8 b) 1 2 3 4 5 c) 0 1 2 3 4 5 6 7 8 9 d) 1 2 3 4 5 6 7 8 9 10

53. What is the output of this program?

```
1.            class variable_scope {  
2.                public static void main(String args[])  
3.                {  
4.                    int x;  
5.                    x = 5;  
6.                    {  
7.                        int y = 6;  
8.                        System.out.print(x + " " + y);  
9.                    }  
10.                System.out.println(x + " " + y);  
11.                }  
12.                }
```

a) 5 6 5 6 b) 5 6 5 c) Runtime error d) Compilation error

54. Which of these is incorrect string literal?

a) "Hello World" b) "Hello\nWorld" c) "\Hello World" d) "Helloworld"

55. What is the output of this program?

```
1.            class dynamic_initialization {  
2.                public static void main(String args[])  
3.                {  
4.                    double a, b;  
5.                    a = 3.0;  
6.                    b = 4.0;  
7.                        double c = Math.sqrt(a * a + b * b);  
8.                        System.out.println(c);  
9.                    }  
10.                }
```

a) 5.0 b) 25.0 c) 7.0 d) Compilation Error

55. Which of these operators is used to allocate memory to array variable in Java?

a) malloc b) alloc c) new d) new malloc

56. Which of these is an incorrect array declaration?

a) `int arr[] = new int[5]` b) `int [] arr = new int[5]` c) `int arr[] arr = new int[5]` d) `int arr[] = int [5]`  
new

57. What will this code print?

```
int arr[] = new int [5];
```

```
System.out.print(arr);
```

a) 0 b) value stored in `arr[0]`. c) 00000 d) Garbage value

58. Which of these is an incorrect Statement?

a) It is necessary to use new operator to initialize an array.

b) Array can be initialized using comma separated expressions surrounded by curly braces.

c) Array can be initialized when they are declared.

d) None of the mentioned

59. Which of these is necessary to specify at time of array initialization?

a) Row b) Column c) Both Row and Column d) None of the mentioned

60. What is the output of this program?

```
1.      class array_output {  
2.          public static void main(String args[])  
3.      {  
4.          int array_variable [] = new int[10];  
5.          for (int i = 0; i < 10; ++i) {
```

```

6.         array_variable[i] = i;
7.         System.out.print(array_variable[i] + " ");
8.         i++;
9.     }
10.    }
11.    }

```

a) 0 2 4 6 8 b) 1 3 5 7 9 c) 0 1 2 3 4 5 6 7 8 9 d) 1 2 3 4 5 6 7 8 9 10

61. What is the output of this program?

```

1.     class multidimension_array {
2.         public static void main(String args[])
3.         {
4.             int arr[][] = new int[3][];
5.             arr[0] = new int[1];
6.             arr[1] = new int[2];
7.             arr[2] = new int[3];
8.             int sum = 0;
9.             for (int i = 0; i < 3; ++i)
10.                for (int j = 0; j < i + 1; ++j)
11.                    arr[i][j] = j + 1;
12.                for (int i = 0; i < 3; ++i)
13.                    for (int j = 0; j < i + 1; ++j)
14.                        sum += arr[i][j];
15.                System.out.print(sum);
16.            }
17.        }

```

a) 11 b) 10 c) 13 d) 14

62. What is the output of this program?

```

1.     class evaluate {
2.         public static void main(String args[])

```

```

3.         {
4.             int arr[] = new int[] {0 , 1, 2, 3, 4, 5, 6, 7, 8, 9};
5.             int n = 6;
6.             n = arr[arr[n] / 2];
7.             System.out.println(arr[n] / 2);
8.         }
9.     }

```

a) 3 b) 0 c) 6 d) 1

63. What is the output of this program?

```

1.     class array_output {
2.         public static void main(String args[])
3.         {
4.             char array_variable [] = new char[10];
5.             for (int i = 0; i < 10; ++i) {
6.                 array_variable[i] = 'i';
7.                 System.out.print(array_variable[i] + "");
8.             }
9.         }
10.    }

```

a) 1 2 3 4 5 6 7 8 9 10 b) 0 1 2 3 4 5 6 7 8 9 10 c) i j k l m n o p q r d) i i i i i i i i i i

64. Which of the following can be operands of arithmetic operators?

a) Numeric b) Boolean c) Characters d) Both Numeric & Characters

65. Modulus operator, %, can be applied to which of these?

a) Integers b) Floating – point numbers c) Both Integers and floating – point numbers. d) None of the mentioned

66. With  $x = 0$ , which of the following are legal lines of Java code for changing the value of  $x$  to 1?

1.  $x++$ ;

2.  $x = x + 1$ ;

3.  $x += 1$ ;

4.  $x =+ 1$ ;

a) 1, 2 & 3 b) 1 & 4 c) 1, 2, 3 & 4 d) 3 & 2

67. Decrement operator,  $-$ , decreases value of variable by what number?

a) 1 b) 2 c) 3 d) 4

68. Which of these statements are incorrect?

a) Assignment operators are more efficiently implemented by Java run-time system than their equivalent long forms.

b) Assignment operators run faster than their equivalent long forms.

c) Assignment operators can be used only with numeric and character data type.

d) None

69. What is the output of this program?

```
1.      class increment {
2.          public static void main(String args[])
3.          {
4.              double var1 = 1 + 5;
5.              double var2 = var1 / 4;
6.              int var3 = 1 + 5;
7.              int var4 = var3 / 4;
8.              System.out.print(var2 + " " + var4);
9.
10.         }
11.     }
```

a) 1 1 b) 0 1 c) 1.5 1 d) 1.5 1.0

70. What is the output of this program?

```
1.      class Modulus {
2.          public static void main(String args[])
3.          {
4.              double a = 25.64;
5.              int b = 25;
6.              a = a % 10;
7.              b = b % 10;
8.              System.out.println(a + " " + b);
9.          }
10.     }
```

a) 5.6400000000000001 5 b) 5.6400000000000001 5.0 c) 5 5 d) 5 5.6400000000000001

71. What is the output of this program?

```
1.      class increment {
2.          public static void main(String args[])
3.          {
4.              int g = 3;
5.              System.out.print(++g * 8);
6.          }
7.     }
```

a) 25 b) 24 c) 32 d) 33

72. What is the output of this program?

```
1.      class Output {
2.          public static void main(String args[])
3.          {
4.              int x , y;
5.              x = 10;
```

```

6.         x++;
7.         --x;
8.         y = x++;
9.         System.out.println(x + " " + y);
10.        }
11.        }

```

a) 11 11 b) 10 10 c) 11 10 d) 10 11

73. What is the output of this program?

```

1.         class Output {
2.             public static void main(String args[])
3.             {
4.                 int a = 1;
5.                 int b = 2;
6.                 int c;
7.                 int d;
8.                 c = ++b;
9.                 d = a++;
10.                c++;
11.                b++;
12.                ++a;
13.                System.out.println(a + " " + b + " " + c);
14.            }
15.        }

```

a) 3 2 4 b) 3 2 3 c) 2 3 4 d) 3 4 4

74. Which of these is not a bitwise operator?

a) & b) &= c) |= d)

75. Which operator is used to invert all the digits in binary representation of a number?

a) ~ b) <<< c) >>> d) ^

76. On applying Left shift operator, <<, on an integer bits are lost one they are shifted past which position bit?

a) 1 b) 32 c) 33 d)

77. Which right shift operator preserves the sign of the value?

a) << b) >> c) <<= d) >>=

78. Which of these statements are incorrect?

a) The left shift operator, <<, shifts all of the bits in a value to the left specified number of times.

b) The right shift operator, >>, shifts all of the bits in a value to the right specified number of times.

c) The left shift operator can be used as an alternative to multiplying by 2.

d) The right shift operator automatically fills the higher order bits with 0.

79. What is the output of this program?

```
1.      class bitwise_operator {
2.          public static void main(String args[])
3.          {
4.              int var1 = 42;
5.              int var2 = ~var1;
6.              System.out.print(var1 + " " + var2);
7.          }
8.      }
```

a) 42 42 b) 43 43 c) 42 -43 d) 42 43

80. What is the output of this program?

```
1.      class bitwise_operator {
2.          public static void main(String args[])
3.          {
4.              int a = 3;
5.              int b = 6;
6.              int c = a | b;
7.              int d = a & b;
```

```
8.         System.out.println(c + " " + d);
9.         }
10.        }
```

a) 7 2 b) 7 7 c) 7 5 d) 5 2

81. What is the output of this program?

```
1.         class leftshift_operator {
2.             public static void main(String args[])
3.             {
4.                 byte x = 64;
5.                 int i;
6.                 byte y;
7.                 i = x << 2;
8.                 y = (byte) (x << 2)
9.                 System.out.print(i + " " + y);
10.            }
11.        }
```

a) 0 64 b) 64 0 c) 0 256 d) 256 0

82. What is the output of this program?

```
1.         class rightshift_operator {
2.             public static void main(String args[])
3.             {
4.                 int x;
5.                 x = 10;
6.                 x = x >> 1;
7.                 System.out.println(x);
8.             }
9.         }
```

a) 10 b) 5 c) 2 d) 20

83. What is the output of this program?

```
1.      class Output {
2.          public static void main(String args[])
3.          {
4.              int a = 1;
5.              int b = 2;
6.              int c = 3;
7.              a |= 4;
8.              b >>= 1;
9.              c <<= 1;
10.             a ^= c;
11.             System.out.println(a + " " + b + " " + c);
12.         }
13.     }
```

a) 3 1 6 b) 2 2 3 c) 2 3 4 d) 3 3 6

84. What is the output of relational operators?

a) Integer b) Boolean c) Characters d) Double

85. Which of these is returned by greater than, <, and equal to, ==, operator?

a) Integers b) Floating - point numbers c) Boolean d) None of the

86 Which of the following operators can operate on a boolean variable? 1. && 2. == 3. ? 4. +=

a) 3 & 2 b) 1 & 4 c) 1, 2 & 4 d) 1, 2 & 3

87. Which of these operators can skip evaluating right hand operand?

a) ! b) | c) & d) &&

88. Which of these statement is correct?

a) true and false are numeric values 1 and 0. b) true and false are numeric values 0 and 1. c) true is any non zero value and false is 0. d) true and false are non numeric values.

89. What is the output of this program?

```
1.      class Relational_operator {
2.          public static void main(String args[])
```

```

3.     {
4.         int var1 = 5;
5.         int var2 = 6;
6.         System.out.print(var1 > var2);
7.     }
8.     }

```

a) 1 b) 0 c) true d) false

90. What is the output of this program?

```

1.     class bool_operator {
2.         public static void main(String args[])
3.         {
4.             boolean a = true;
5.             boolean b = !true;
6.             boolean c = a | b;
7.             boolean d = a & b;
8.             boolean e = d ? b : c;
9.             System.out.println(d + " " + e);
10.        }
11.    }

```

a) false false b) true ture c) true false d) false true

91. What is the output of this program?

```

1.     class ternary_operator {
2.         public static void main(String args[])
3.         {
4.             int x = 3;
5.             int y = ~ x;
6.             int z;
7.             z = x > y ? x : y;
8.             System.out.print(z);

```

```
9.      }
10.     }
```

a) 0 b) 1 c) 3 d) -4

92. What is the output of this program?

```
1.      class Output {
2.          public static void main(String args[])
3.          {
4.              int x , y = 1;
5.              x = 10;
6.              if (x != 10 && x / 0 == 0)
7.                  System.out.println(y);
8.              else
9.                  System.out.println(++y);
10.         }
11.     }
```

a) 1 b) 2 c) Runtime error owing to division by zero in if condition. d) Unpredictable behavior of program.

93. What is the output of this program?

```
1.      class Output {
2.          public static void main(String args[])
3.          {
4.              boolean a = true;
5.              boolean b = false;
6.              boolean c = a ^ b;
7.              System.out.println(!c);
8.          }
9.     }
```

a) 0 b) 1 c) false d) true

94. Which of these selection statements test only for equality?

a) if b) switch c) if & switch d) None of the mentioned

95. Which of these are selection statements in Java?

a) if() b) for() c) continue d) break

96. Which of the following loops will execute the body of loop even when condition controlling the loop is initially false?

a) do-while b) while c) for d) None of the mentioned

97. Which of these jump statements can skip processing remainder of code in its body for a particular iteration?

a) break b) return c) exit d) continue

98. Which of these statement is correct?

a) switch statement is more efficient than a set of nested ifs.

b) two case constants in the same switch can have identical values.

c) switch statement can only test for equality, whereas if statement can evaluate any type of boolean expression.

d) it is possible to create a nested switch statements.

99. What is the output of this program?

```
1.      class selection_statements {
2.          public static void main(String args[])
3.          {
4.              int var1 = 5;
5.              int var2 = 6;
6.              if ((var2 = 1) == var1)
7.                  System.out.print(var2);
8.          else
```

```
9.         System.out.print(++var2);
10.        }
11.        }
```

a) 1 b) 2 c) 3 d) 4

100. What is the output of this program?

```
1.         class comma_operator {
2.             public static void main(String args[])
3.             {
4.                 int sum = 0;
5.                 for (int i = 0, j = 0; i < 5 & j < 5; ++i, j = i + 1)
6.                     sum += i;
7.                 System.out.println(sum);
8.             }
9.         }
```

a) 5 b) 6 c) 14 d) compilation error

101. What is the output of this program?

```
1.         class jump_statments {
2.             public static void main(String args[])
3.             {
4.                 int x = 2;
5.                 int y = 0;
6.                 for (; y < 10; ++y) {
7.                     if (y % x == 0)
8.                         continue;
9.                     else if (y == 8)
10.                        break;
11.                    else
12.                        System.out.print(y + " ");
13.                }
```

14.            }  
15.            }

a) 1 3 5 7 b) 2 4 6 8 c) 1 3 5 7 9 d) 1 2 3 4 5 6 7 8 9

102. What is the output of this program?

```
1.            class Output {  
2.                public static void main(String args[])  
3.                {  
4.                    int x, y = 1;  
5.                    x = 10;  
6.                    if (x != 10 && x / 0 == 0)  
7.                        System.out.println(y);  
8.                    else  
9.                        System.out.println(++y);  
10.                }  
11.            }
```

a) 1 b) 2 c) Runtime error owing to division by zero in if condition. d) Unpredictable behavior of program.

103. What is the output of this program?

```
1.            class Output {  
2.                public static void main(String args[])  
3.                {  
4.                    int a = 5;  
5.                    int b = 10;  
6.                    first: {  
7.                        second: {  
8.                            third: {  
9.                                if (a == b >> 1)  
10.                                    break second;  
11.                                }  
                          }  
                  }  
          }
```

```
12.         System.out.println(a);
13.         }
14.         System.out.println(b);
15.         }
16.     }
17. }
```

a) 5 10 b) 10 5 c) 5 d) 10

104. What is stored in the object obj in following lines of code?

box obj;

a) Memory address of allocated memory of object. b) NULL c) Any arbitrary pointer d) Garbage

105. Which of these keywords is used to make a class?

a) class b) struct c) int d) None of the mentioned

106. Which of the following is a valid declaration of an object of class Box?

a) Box obj = new Box(); b) Box obj = new Box; c) obj = new Box(); d) new Box obj;

107. Which of these operators is used to allocate memory for an object?

a) malloc b) alloc c) new d) give

108. Which of these statements is incorrect?

a) Every class must contain a main() method. b) Applets do not require a main() method at all.  
c) There can be only one main() method in a program. d) main() method must be made public.

109. What is the output of this program?

```
1.     class main_class {
2.         public static void main(String args[])
3.         {
4.             int x = 9;
5.             if (x == 9) {
```

```
6.         int x = 8;
7.         System.out.println(x);
8.     }
9. }
10. }
```

a) 9 b) 8 c) Compilation error d) Runtime error

110. Which of the following statements is correct?

- a) Public method is accessible to all other classes in the hierarchy
- b) Public method is accessible only to subclasses of its parent class
- c) Public method can only be called by object of its class.
- d) Public method can be accessed by calling object of the public class.

111. What is the output of this program?

```
1.     class box {
2.         int width;
3.         int height;
4.         int length;
5.     }
6.     class mainclass {
7.         public static void main(String args[])
8.         {
9.             box obj = new box();
10.            obj.width = 10;
11.            obj.height = 2;
12.            obj.length = 10;
13.            int y = obj.width * obj.height * obj.length;
14.            System.out.print(y);
15.        }
16.    }
```

a) 12 b) 200 c) 400 d) 100

112. What is the output of this program?

```
1.      class box {
2.          int width;
3.          int height;
4.          int length;
5.      }
6.      class mainclass {
7.          public static void main(String args[])
8.          {
9.              box obj1 = new box();
10.             box obj2 = new box();
11.             obj1.height = 1;
12.             obj1.length = 2;
13.             obj1.width = 1;
14.             obj2 = obj1;
15.             System.out.println(obj2.height);
16.         }
17.     }
```

a) 1 b) 2 c) Runtime error d) Garbage value

113. What is the output of this program?

```
1.      class box {
2.          int width;
3.          int height;
4.          int length;
5.      }
6.      class mainclass {
7.          public static void main(String args[])
8.          {
```

```
9.         box obj = new box();
10.        System.out.println(obj);
11.        }
12.        }
```

a) 0 b) 1 c) Runtime error d) Garbage value

114. What is the return type of a method that does not returns any value?

a) int b) float c) void d) double

115. What is the process of defining more than one method in a class differentiated by method signature?

a) Function overriding b) Function overloading c) Function doubling d) None of the mentioned

116. Which of the following is a method having same name as that of it's class?

a) finalize b) delete c) class d) constructor

117. Which method can be defined only once in a program?

a) main method b) finalize method c) static method d) private method

118. Which of these statement is incorrect?

- a) All object of a class are allotted memory for the all the variables defined in the class.
- b) If a function is defined public it can be accessed by object of other class by inheritance.
- c) main() method must be made public.
- d) All object of a class are allotted memory for the methods defined in the class.

119. What is the output of this program?

```
1.         class box {
2.             int width;
3.             int height;
4.             int length;
5.             int volume;
```

```

6.         void volume(int height, int length, int width) {
7.             volume = width*height*length;
8.         }
9.     }
10.    class Prameterized_method{
11.        public static void main(String args[])
12.        {
13.            box obj = new box();
14.            obj.height = 1;
15.            obj.length = 5;
16.            obj.width = 5;
17.            obj.volume(3,2,1);
18.            System.out.println(obj.volume);
19.        }
20.    }

```

a) 0 b) 1 c) 6 d) 25

120. What is the output of this program?

```

1.    class equality {
2.        int x;
3.        int y;
4.        boolean isequal(){
5.            return(x == y);
6.        }
7.    }
8.    class Output {
9.        public static void main(String args[])
10.       {
11.           equality obj = new equality();
12.           obj.x = 5;
13.           obj.y = 5;

```

```
14.         System.out.println(obj.isequal());
15.     }
16. }
```

a) false b) true c) 0 d) 1

121. What is the output of this program?

```
1.     class box {
2.         int width;
3.         int height;
4.         int length;
5.         int volume;
6.         void volume() {
7.             volume = width*height*length;
8.         }
9.     }
10.    class Output {
11.        public static void main(String args[])
12.        {
13.            box obj = new box();
14.            obj.height = 1;
15.            obj.length = 5;
16.            obj.width = 5;
17.            obj.volume();
18.            System.out.println(obj.volume);
19.        }
20.    }
```

a) 0 b) 1 c) 25 d) 26

122. What is the output of this program?

```
1.     class Output {
2.         static void main(String args[])
```

```
3.      {
4.      int x , y = 1;
5.      x = 10;
6.      if (x != 10 && x / 0 == 0)
7.          System.out.println(y);
8.      else
9.          System.out.println(++y);
10.     }
11.     }
```

a) 1 b) 2 c) Runtime Error d) Compilation Error

123. What is the output of this program?

```
1.      class area {
2.      int width;
3.      int length;
4.      int volume;
5.      area() {
6.      width=5;
7.      length=6;
8.      }
9.      void volume() {
10.     volume = width*length*height;
11.     }
12.     }
13.     class cons_method {
14.     public static void main(String args[])
15.     {
16.     area obj = new area();
17.     obj.volume();
18.     System.out.println(obj.volume);
19.     }
```

20.                    }

a) 0 b) 1 c) 30 d) error

124. What is the return type of Constructors?

a) int b) float c) void d) None of the mentioned

125. Which keyword is used by method to refer to the object that invoked it?

a) import b) catch c) abstract d) this

126. Which of the following is a method having same name as that of its class?

a) finalize b) delete c) class d) constructor

127. Which operator is used by Java run time implementations to free the memory of an object when it is no longer needed?

a) delete b) free c) new d) None of the mentioned

128. Which function is used to perform some action when the object is to be destroyed?

a) finalize() b) delete() c) main() d) None of the mentioned

129. What is the output of this program?

```
1.      class box {
2.          int width;
3.          int height;
4.          int length;
5.          int volume;
6.          box() {
7.              width = 5;
8.              height = 5;
9.              length = 6;
10.         }
11.         void volume() {
```

```

12.         volume = width*height*length;
13.     }
14. }
15. class constructor_output {
16.     public static void main(String args[])
17.     {
18.         box obj = new box();
19.         obj.volume();
20.         System.out.println(obj.volume);
21.     }
22. }

```

a) 100 b) 150 c) 200 d) 250

130. What is the output of this program?

```

1.     class equality {
2.         int x;
3.         int y;
4.         boolean isequal() {
5.             return(x == y);
6.         }
7.     }
8.     class Output {
9.         public static void main(String args[])
10.        {
11.            equality obj = new equality();
12.            obj.x = 5;
13.            obj.y = 5;
14.            System.out.println(obj.isequal);    }
15.        }

```

a) false b) true c) 0 d) 1

131. What is the output of this program?

```
1.      class box {
2.          int width;
3.          int height;
4.          int length;
5.          int volume;
6.          void finalize() {
7.              volume = width*height*length;
8.              System.out.println(volume);
9.          }
10.         protected void volume() {
11.             volume = width*height*length;
12.             System.out.println(volume);
13.         }
14.     }
15.     class Output {
16.         public static void main(String args[])
17.         {
18.             box obj = new box();
19.             obj.volume();
20.         }
21.     }
```

a) 150 b) 200 c) Runtime error d) Compilation error

132. Which of the following statements are incorrect?

- a) Default constructor is called at the time of declaration of the object if a constructor has not been defined.
- b) Constructor can be parameterized.
- c) finalize() method is called when an object goes out of scope and is no longer needed.
- d) finalize() method must be declared protected.

133. What is the output of this program?

```
1.      class area {
2.          int width;
3.          int length;
4.          int area;
5.          void area(int width, int length) {
6.              this.width = width;
7.              this.length = length;
8.          }
9.
10.     }
11.     class Output {
12.         public static void main(String args[])
13.         {
14.             area obj = new area();
15.             obj.area(5 , 6);
16.             System.out.println(obj.length + " " + obj.width);
17.         }
18.     }
```

a) 0 0 b) 5 6 c) 6 5 d) 5 5

134 What is process of defining two or more methods within same class that have same name but different parameters declaration?

a) method overloading b) method overriding c) method hiding d) None of the mentioned

135. Which of these can be overloaded?

a) Methods b) Constructors c) All of the mentioned d) None of the mentioned

136. Which of these is correct about passing an argument by call-by-value process?

a) Copy of argument is made into the formal parameter of the subroutine.

b) Reference to original argument is passed to formal parameter of the subroutine.

c) Copy of argument is made into the formal parameter of the subroutine and changes made on parameters of subroutine have effect on original argument.

d) Reference to original argument is passed to formal parameter of the subroutine and changes made on parameters of subroutine have effect on original argument.

137. What is the process of defining a method in terms of itself, that is a method that calls itself?

a) Polymorphism b) Abstraction c) Encapsulation d) Recursion

138. Which of the following statements are incorrect?

a) Default constructor is called at the time of declaration of the object if a constructor has not been defined.

b) Constructor can be parameterized.

c) finalize() method is called when a object goes out of scope and is no longer needed.

d) finalize() method must be declared protected.

139. What is the output of this program?

```
1.      class overload {
2.          int x;
3.          int y;
4.          void add(int a) {
5.              x = a + 1;
6.          }
7.          void add(int a, int b){
8.              x = a + 2;
9.          }
10.     }
11.     class Overload_methods {
12.         public static void main(String args[])
13.         {
14.             overload obj = new overload();
15.             int a = 0;
```

```
16.         obj.add(6);
17.         System.out.println(obj.x);
18.     }
19. }
```

a) 5 b) 6 c) 7 d) 8

140. What is the output of this program?

```
1.     class overload {
2.         int x;
3.         int y;
4.         void add(int a){
5.             x = a + 1;
6.         }
7.         void add(int a , int b){
8.             x = a + 2;
9.         }
10.    }
11.    class Overload_methods {
12.        public static void main(String args[])
13.        {
14.            overload obj = new overload();
15.            int a = 0;
16.            obj.add(6, 7);
17.            System.out.println(obj.x);
18.        }
19.    }
```

a) 6 b) 7 c) 8 d) 9

141. What is the output of this program?

```
1.     class overload {
2.         int x;
```

```

3.         double y;
4.     void add(int a , int b) {
5.         x = a + b;
6.     }
7.     void add(double c , double d){
8.         y = c + d;
9.     }
10.    overload() {
11.        this.x = 0;
12.        this.y = 0;
13.    }
14. }
15. class Overload_methods {
16.     public static void main(String args[])
17.     {
18.         overload obj = new overload();
19.         int a = 2;
20.         double b = 3.2;
21.         obj.add(a, a);
22.         obj.add(b, b);
23.         System.out.println(obj.x + " " + obj.y);
24.     }
25. }

```

a) 6 6 ) 6.4 6.4 c) 6.4 6 d) 4 6.4

142. What is the output of this program?

```

1.     class test {
2.         int a;
3.         int b;
4.         void meth(int i, int j) {
5.             i *= 2;

```

```

6.         j /= 2;
7.     }
8. }
9.     class Output {
10.        public static void main(String args[])
11.        {
12.            test obj = new test();
13.            int a = 10;
14.            int b = 20;
15.            obj.meth(a , b);
16.            System.out.println(a + " " + b);
17.        }
18.    }

```

a) 10 20 b) 20 10 c) 20 40 d) 40 20

143. What is the output of this program?

```

1.     class test {
2.         int a;
3.         int b;
4.         test(int i, int j) {
5.             a = i;
6.             b = j;
7.         }
8.         void meth(test o) {
9.             o.a *= 2;
10.            O.b /= 2;
11.        }
12.    }
13.    class Output {
14.        public static void main(String args[])
15.        {

```

```
16.         test obj = new test(10 , 20);
17.         obj.meth(obj);
18.         System.out.println(obj.a + " " + obj.b);
19.     }
20. }
```

a) 10 20 b) 20 10 c) 20 40 d) 40 20

144. Which of these access specifiers must be used for main() method?

a) private b) public c) protected d) None of the mentioned

145. Which of these is used to access member of class before object of that class is created?

a) public b) private c) static d) protected

146. Which of these is used as default for a member of a class if no access specifier is used for it?

a) private b) public c) public, within its own package d) protected

147. What is the process by which we can control what parts of a program can access the members of a class?

a) Polymorphism b) Abstraction c) Encapsulation d) Recursion

148. Which of the following statements are incorrect?

a) public members of class can be accessed by any code in the program.

b) private members of class can only be accessed by other members of the class.

c) private members of class can be inherited by a sub class, and become protected members in sub class.

d) protected members of a class can be inherited by a sub class, and become private members of the sub class.

149. What is the output of this program?

```
1.         class access{
```

```

2.      public int x;
3.          private int y;
4.      void cal(int a, int b){
5.          x = a + 1;
6.          y = b;
7.      }
8.  }
9.      class access_specifier {
10.         public static void main(String args[])
11.         {
12.             access obj = new access();
13.             obj.cal(2, 3);
14.             System.out.println(obj.x + " " + obj.y);
15.         }
16.     }

```

a) 3 3 b) 2 3 c) Runtime Error d) Compilation Error

150. What is the output of this program?

```

1.      class access{
2.          public int x;
3.          private int y;
4.      void cal(int a, int b){
5.          x = a + 1;
6.          y = b;
7.      }
8.      void print() {
9.          system.out.println(" " + y);
10.     }
11. }
12. class access_specifier {
13.     public static void main(String args[])

```

```

14.     {
15.         access obj = new access();
16.         obj.cal(2, 3);
17.         System.out.println(obj.x);
18.         obj.print();
19.     }
20.     }

```

a) 2 3 b) 3 3 c) Runtime Error d) Compilation Error

151 What is the output of this program?

```

1.     class static_out {
2.         static int x;
3.         static int y;
4.         void add(int a, int b){
5.             x = a + b;
6.             y = x + b;
7.         }
8.     }
9.     class static_use {
10.        public static void main(String args[])
11.        {
12.            static_out obj1 = new static_out();
13.            static_out obj2 = new static_out();
14.            int a = 2;
15.            obj1.add(a, a + 1);
16.            obj2.add(5, a);
17.            System.out.println(obj1.x + " " + obj2.y);
18.        }
19.    }

```

a) 7 7 b) 6 6 c) 7 9 d) 9 7

152. Which of these access specifier must be used for class so that it can be inherited by another sub class?

a) public b) private c) protected d) None of the mentioned

153. What is the output of this program?

```
1.      class test {
2.          int a;
3.          int b;
4.          test(int i, int j) {
5.              a = i;
6.              b = j;
7.          }
8.          void meth(test o) {
9.              o.a *= 2;
10.             O.b /= 2;
11.         }
12.     }
13.     class Output {
14.         public static void main(String args[])
15.         {
16.             test obj = new test(10 , 20);
17.             obj.meth(obj);
18.             System.out.println(obj.a + " " + obj.b);    }
19.     }
```

a) 10 20 b) 20 10 c) 20 40 d) 40 20

154. Arrays in Java are implemented as?

a) class b) object c) variable d) None of the mentioned

155. Which of these keywords is used to prevent content of a variable from being modified?

- a) final b) last c) constant d) static

156. Which of these cannot be declared static?

- a) class b) object c) variable d) method

157. Which of the following statements are incorrect?

- a) static methods can call other static methods only.  
b) static methods must only access static data.  
c) static methods can not refer to this or super in any way.  
d) when object of class is declared, each object contains its own copy of static variables.

158. Which of the following statements are incorrect?

- a) Variables declared as final occupy memory.  
b) final variable must be initialized at the time of declaration.  
c) Arrays in java are implemented as an object.  
d) All arrays contain an attribute-length which contains the number of elements stored in the array.

159. Which of these methods must be made static?

- a) main() b) delete() c) run() d) finalize()

160. What is the output of this program?

```
1.      class access{  
2.          public int x;  
3.          static int y;  
4.          void cal(int a, int b){  
5.              x += a ;  
6.              y += b;  
7.          }  
8.      }
```

```

9.      class static_specifier {
10.         public static void main(String args[])
11.         {
12.             access obj1 = new access();
13.             access obj2 = new access();
14.             obj1.x = 0;
15.             obj1.y = 0;
16.             obj1.cal(1, 2);
17.             obj2.x = 0;
18.             obj2.cal(2, 3);
19.             System.out.println(obj1.x + " " + obj2.y);
20.         }
21.     }

```

a) 1 2 b) 2 3 c) 3 2 d) 1 5

161. What is the output of this program?

```

1.      class access{
2.         static int x;
3.         void increment(){
4.             x++;
5.         }
6.     }
7.      class static_use {
8.         public static void main(String args[])
9.         {
10.             access obj1 = new access();
11.             access obj2 = new access();
12.             obj1.x = 0;
13.             obj1.increment();
14.             obj2.increment();
15.             System.out.println(obj1.x + " " + obj2.x);

```

```
16.         }
17.         }
```

a) 1 2 b) 1 1 c) 2 2 d) Compilation Error

162. What is the output of this program?

```
1.         class static_out {
2.             static int x;
3.             static int y;
4.             void add(int a , int b){
5.                 x = a + b;
6.                 y = x + b;
7.             }
8.         }
9.         class static_use {
10.            public static void main(String args[])
11.            {
12.                static_out obj1 = new static_out();
13.                static_out obj2 = new static_out();
14.                int a = 2;
15.                obj1.add(a, a + 1);
16.                obj2.add(5, a);
17.                System.out.println(obj1.x + " " + obj2.y);
18.            }
19.        }
```

a) 7 7 b) 6 6 c) 7 9 d) 9 7

163. What is the output of this program?

```
1.         class Output {
2.             public static void main(String args[])
3.             {
4.                 int arr[] = {1, 2, 3, 4, 5};
```

```
5.         for ( int i = 0; i < arr.length - 2; ++i)
6.             System.out.println(arr[i] + " ");
7.         }
8.     }
```

a) 1 2 b) 1 2 3 c) 1 2 3 4 d) 1 2 3 4 5

164. What is the output of this program?

```
1.     class Output {
2.         public static void main(String args[])
3.         {
4.             int a1[] = new int[10];
5.             int a2[] = {1, 2, 3, 4, 5};
6.             System.out.println(a1.length + " " + a2.length);
7.         }
8.     }
```

a) 10 5 b) 5 10 c) 0 10 d) 0 5

165. String in Java is a?

a) class b) object c) variable d) character array

166. Which of these method of String class is used to obtain character at specified index?

a) char() b) Charat() c) charat() d) charAt()

167. Which of these keywords is used to refer to member of base class from a sub class?

a) upper b) super c) this d) None of the mentioned

168. Which of these method of String class can be used to test to strings for equality?

a) isequal() b) isequals() c) equal() d) equals()

169. Which of the following statements are incorrect?

a) String is a class. b) Strings in java are mutable. c) Every string is an object of class String.

d) Java defines a peer class of String, called StringBuffer, which allows string to be altered.

170. What is the output of this program?

```
1.      class string_demo {  
2.          public static void main(String args[])  
3.          {  
4.              String obj = "I" + "like" + "Java";  
5.              System.out.println(obj);  
6.          }  
7.      }
```

a) I b) like c) Java d) IlikeJava

171. What is the output of this program?

```
1.      class string_class {  
2.          public static void main(String args[])  
3.          {  
4.              String obj = "I LIKE JAVA";  
5.              System.out.println(obj.charAt(3));  
6.          }  
7.      }
```

a) I b) L c) K d) E

172. What is the output of this program?

```
1.      class string_class {  
2.          public static void main(String args[])  
3.          {  
4.              String obj = "I LIKE JAVA";  
5.              System.out.println(obj.length());  
6.          }  
7.      }
```

a) 9 b) 10 c) 11 d) 12

173. What is the output of this program?

```
1.      class string_class {
2.          public static void main(String args[])
3.          {
4.              String obj = "hello";
5.              String obj1 = "world";
6.              String obj2 = obj;
7.              obj2 = " world";
8.              System.out.println(obj + " " + obj2);
9.          }
10.     }
```

a) hello hello b) world world c) hello world d) world hello

174. What is the output of this program?

```
1.      class string_class {
2.          public static void main(String args[])
3.          {
4.              String obj = "hello";
5.              String obj1 = "world";
6.              String obj2 = "hello";
7.              System.out.println(obj.equals(obj1) + " " + obj.equals(obj2));
8.          }
9.     }
```

a) false false b) true true c) true false d) false true

## Unit-2

1. \_\_\_\_\_ must always be the first statement executed inside a subclass constructor.

2. Main use \_\_\_\_\_ statement to exit from current method.
3. \_\_\_\_\_ is a constructor that replicates or duplicates an existing object.
4. \_\_\_\_\_ classes cannot declare constructors.
5. Which one of the following defines a legal abstract class?
  - a) abstract class Vehicle {<br> abstract void display(); }
  - b) class Vehicle {<br> abstract void display(); }
  - c) abstract Vehicle {<br> abstract void display(); }
  - d) class abstract Vehicle {<br> abstract void display(); }
6. \_\_\_\_\_ is used to convert the one type value to another.
7. \_\_\_\_\_ method cannot be overridden.
8. \_\_\_\_\_ implies that a sub class object can be substituted for an object of any of its super class with no observable difference in behavior.
9. \_\_\_\_\_ is a phenomenon where a subclass is promoted to a super class and hence becomes more general generalization needs up-casting.
  - a. inheritance b. generalization c. polymorphism d. event
10. Abstract class and abstract method should be declared by using the keyword
  - a. abstract b. super c. Class d. none
11. writing two or more methods in super and sub classes such that the methods have the same name and same signature is called
  - a. method overriding. B.operator overloading c. overloading d.none
12. The Polymorphism exhibited at run time is called
  - a. dynamic Polymorphism b. static polymorphism c. a&b d none
13. The keyword \_\_\_\_\_ signifies that the properties of the super class name are extended to the sub class name
  - a.extends b. super c. subclass d class
14. Super can be used to refer to super class method as
  - a.super.class() b.super.method() c. a&b d. none

15. Which of these keyword must be used to inherit a class?

a) super b) this c )

extent

d) extends

16. Which of these keywords is used to refer to member of base class from a sub class?

a) upper

b) super

c) this

d) None of the mentioned

17. A class member declared protected becomes member of subclass of which type?

a) public member b) private member

c) protected member

d) static member

18. Which of these is correct way of inheriting class A by class B?

a) class B + class A { }

b) class B inherits class A { }

c) class B extends A { }

d) class B extends class A { }

19. Which of the following statements are incorrect?

a) public members of class can be accessed by any code in the program.

b) private members of class can only be accessed by other members of the class.

c) private members of class can be inherited by a sub class, and become protected members in sub class.

d) protected members of a class can be inherited by a sub class, and become

private members of the sub class.

20. What is the output of this program?

```
class A {
    int i;
    void display() {
        System.out.println(i);
    }
}

class B extends A {
    int j;
    void display() {
        System.out.println(j);
    }
}

class inheritance_demo {
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

- a) 0
- b) 1
- c) 2
- d) Compilation Error

21. What is the output of this program?

```
class A {
```

```

    int i;
}
class B extends A {
    int j;
    void display() {
        super.i = j + 1;
        System.out.println(j + " " + i);
    }
}
class inheritance {
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}

```

- a) 2 2
- b) 3 3
- c) 2 3
- d) 3 2

22. What is the output of this program?

```

class A {
    public int i;
    private int j;
}
class B extends A {
    void display() {
        super.j = super.i + 1;
    }
}

```

```

        System.out.println(super.i + " " + super.j);
    }
}
class inheritance {
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}

```

- a) 2 2
- b) 3 3
- c) Runtime Error
- d) Compilation Error

24. What is the output of this program?

```

class A {
    public int i;
    public int j;
    A() {
        i = 1;
        j = 2;
    }
}
class B extends A {
    int a;
    B() {
        super();
    }
}

```

```

    }
    class super_use {
        public static void main(String args[])
        {
            B obj = new B();
            System.out.println(obj.i + " " + obj.j)
        }
    }

```

- a) 1 2
- b) 2 1
- c) Runtime Error
- d) Compilation Error

25. What is the output of this program?

```

class A {
    public int i;
    protected int j;
}

class B extends A {
    int j;
    void display() {
        super.j = 3;
        System.out.println(i + " " + j);
    }
}

class Output {
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
    }
}

```

```
        obj.display();
    }
}
```

- a) 1 2
- b) 2 1
- c) 1 3
- d) 3 1

26 Which of these keyword can be used in subclass to call the constructor of superclass?

- a) super
- b) this
- c) extent
- d) extends

27 .What is the process of defining a method in subclass having same name & type signature as a method in its superclass?

- a) Method overloading
- b) Method overriding
- c) Method hiding
- d) None of the mentioned

28. Which of these keywords can be used to prevent Method overriding?

- a) static
- b) constant
- c) protected
- d) final

29. Which of these is correct way of calling a constructor having no parameters, of superclass A by subclass B?

- a) super(void);

- b) superclass.();
- c) super.A();
- d) super();

30. Which of the following statements are incorrect?

- a) public members of class can be accessed by any code in the program.
- b) private members of class can only be accessed by other members of the class.
- c) private members of class can be inherited by a sub class, and become protected members in sub class.
- d) protected members of a class can be inherited by a sub class, and become private members of the sub class.

31. Which of these is supported by method overriding in Java?

- a) Abstraction
- b) Encapsulation
- c) Polymorphism
- d) None of the mentioned

32. What is the output of this program?

```
class A {
    int i;
    void display() {
        System.out.println(i);
    }
}

class B extends A {
    int j;
    void display() {
        System.out.println(j);
    }
}
```

```
class method_overriding {  
    public static void main(String args[])  
    {  
        B obj = new B();  
        obj.i=1;  
        obj.j=2;  
        obj.display();  
    }  
}
```

- a) 0
- b) 1
- c) 2
- d) Compilation Error

33. What is the output of this program?

```
final class A {  
    int i;  
}  
class B extends A {  
    int j;  
    System.out.println(j + " " + i);  
}  
class inheritance {  
    public static void main(String args[])  
    {  
        B obj = new B();  
        obj.display();  
    }  
}
```

- a) 2 2
- b) 3 3

- c) Runtime Error
- d) Compilation Error

34. What is the output of this program?

```
class A {
    public int i;
    private int j;
}
class B extends A {
    void display() {
        super.j = super.i + 1;
        System.out.println(super.i + " " + super.j);
    }
}
class inheritance {
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

- a) 2 2
- b) 3 3
- c) Runtime Error
- d) Compilation Error

[View Answer](#)

35. What is the output of this program?

```
class A {
    public void display() {
```

```

        System.out.println("A");
    }
}
class B extends A {
    public void display() {
        System.out.println("B");
    }
}
class Dynamic_dispatch {
    public static void main(String args[])
    {
        A obj1 = new A();
        B obj2 = new B();
        A r;
        r = obj1;
        r.display();
        r = obj2;
        r.display();
    }
}

```

- a) A B
- b) B A
- c) Runtime Error
- d) Compilation Error

36. What is the output of this program?

```

class A {
    int i;
    public void display() {
        System.out.println(i);
    }
}

```

```
    }  
    class B extends A {  
        int j;  
        public void display() {  
            System.out.println(j);  
        }  
    }  
    class Dynamic_dispatch {  
        public static void main(String args[])  
        {  
            B obj2 = new B();  
            obj2.i = 1;  
            obj2.j = 2;  
            A r;  
            r = obj2;  
            r.display();  
        }  
    }
```

- a) 1
- b) 2
- c) 3
- d) 4

37. Which of these class is superclass of every class in Java?

- a) String class
- b) Object class
- c) Abstract class
- d) ArrayList class

38. Which of these method of Object class can clone an object?

- a) Objectcopy()

- b) copy()
- c) Object clone()
- d) clone()

39. Which of these method of Object class is used to obtain class of an object at run time?

- a) get()
- b) void getClass()
- c) Class getClass()
- d) None of the mentioned

40. Which of these keywords can be used to prevent inheritance of a class?

- a) super
- b) constant
- c) Class
- d) final

41. Which of these keywords cannot be used for a class which has been declared final?

- a) abstract
- b) extends
- c) abstract and extends
- d) None of the mentioned

42. Which of these class relies upon its subclasses for complete implementation of its methods?

- a) Object class
- b) abstract class
- c) ArrayList class
- d) None of the mentioned

43. What is the output of this program?

```
abstract class A {
    int i;
    abstract void display();
}
class B extends A {
    int j;
    void display() {
        System.out.println(j);
    }
}
class Abstract_demo {
    public static void main(String args[])
    {
        B obj = new B();
        obj.j=2;
        obj.display();
    }
}
```

- a) 0
- b) 2
- c) Runtime Error
- d) Compilation Error

44. What is the output of this program?

```
class A {
    int i;
    int j;
    A() {
        i = 1;
        j = 2;
    }
}
```

```

    }
}
class Output {
    public static void main(String args[])
    {
        A obj1 = new A();
        A obj2 = new A();
        System.out.print(obj1.equals(obj2));
    }
}

```

- a) false
- b) true
- c) 1
- d) Compilation Error

45. What is the output of this program?

```

class Output {
    public static void main(String args[])
    {
        Object obj = new Object();
        System.out.print(obj.getClass());
    }
}

```

- a) Object
- b) class Object
- c) class java.lang.Object
- d) Compilation Error

46. What is the output of this program?

```

class A {
    int i;
}

```

```
        int j;  
        A() {  
            i = 1;  
            j = 2;  
        }  
    }  
    class Output {  
        public static void main(String args[])  
        {  
            A obj1 = new A();  
            System.out.print(obj1.toString());  
        }  
    }
```

- a) true
- b) false
- c) String associated with obj1
- d) Compilation Error

47. Which of these keywords is used to define packages in Java?

- a) pkg
- b) Pkg
- c) package
- d) Package

48. Which of these is a mechanism for naming and visibility control of a class and its content?

- a) Object
- b) Packages
- c) Interfaces
- d) None of the Mentioned.

49. Which of this access specifies can be used for a class so that its members can be accessed by a different class in the same package?

- a) Public
- b) Protected
- c) No Modifier
- d) All of the mentioned

50. Which of these access specifiers can be used for a class so that it's members can be accessed by a different class in the different package?

- a) Public
- b) Protected
- c) Private
- d) No Modifier

51. Which of the following is correct way of importing an entire package 'pkg'?

- a) import pkg.
- b) Import pkg.
- c) import pkg.\*
- d) Import pkg.\*

52. Which of the following is incorrect statement about packages?

- a) Package defines a namespace in which classes are stored.
- b) A package can contain other package within it.
- c) Java uses file system directories to store packages.
- d) A package can be renamed without renaming the directory in which the classes are stored.

53. Which of the following package stores all the standard java classes?

- a) lang
- b) java
- c) util

d) java.packages

54. What is the output of this program?

```
package pkg;
class display {
    int x;
    void show() {
        if (x > 1)
            System.out.print(x + " ");
    }
}
class packages {
    public static void main(String args[]) {
        display[] arr=new display[3];
        for(int i=0;i<3;i++)
            arr[i]=new display();
        arr[0].x = 0;
        arr[1].x = 1;
        arr[2].x = 2;
        for (int i = 0; i < 3; ++i)
            arr[i].show();
    }
}
```

Note : packages.class file is in directory pkg;

- a) 0
- b) 1
- c) 2
- d) 0 1 2

55. What is the output of this program?

```
package pkg;
```

```
class output {  
    public static void main(String args[])  
    {  
        StringBuffer s1 = new StringBuffer("Hello");  
        s1.setCharAt(1, x);  
        System.out.println(s1);  
    }  
}
```

- a) xello
- b) xxxxx
- c) Hxlllo
- d) Hexlo

56. What is the output of this program?

```
package pkg;  
class output {  
    public static void main(String args[])  
    {  
        StringBuffer s1 = new StringBuffer("Hello World");  
        s1.insert(6, "Good ");  
        System.out.println(s1);  
    }  
}
```

Note : Output.class file is not in directory pkg.

- a) HelloGoodWorld
- b) HellGoodoWorld
- c) Compilation error
- d) Runtime error

57. Which of these keywords is used to define interfaces in Java?

- a) interface
- b) Interface
- c) intf
- d) Intf

58. Which of these can be used to fully abstract a class from its implementation?

- a) Objects
- b) Packages
- c) Interfaces
- d) None of the Mentioned.

59. Which of these access specifiers can be used for an interface?

- a) Public
- b) Protected
- c) private
- d) All of the mentioned

60. Which of these keywords is used by a class to use an interface defined previously?

- a) import
- b) Import
- c) implements
- d) Implements

61. Which of the following is correct way of implementing an interface salary by class manager?

- a) class manager extends salary { }
- b) class manager implements salary { }
- c) class manager imports salary { }
- d) None of the mentioned.

62. Which of the following is incorrect statement about packages?

- a) Interfaces specifies what class must do but not how it does.
- b) Interfaces are specified public if they are to be accessed by any code in the program.
- c) All variables in interface are implicitly final and static.
- d) All variables are static and methods are public if interface is defined public.

63. Which of the following package stores all the standard java classes?

- a) lang
- b) java
- c) util
- d) java.packages

64. What is the output of this program?

```
interface calculate {  
    void cal(int item);  
}  
class display implements calculate {  
    int x;  
    public void cal(int item) {  
        x = item * item;  
    }  
}  
class interfaces {  
    public static void main(String args[]) {  
        display arr = new display;  
        arr.x = 0;  
        arr.cal(2);  
        System.out.print(arr.x);  
    }  
}
```

```
}
```

- a) 0
- b) 2
- c) 4
- d) None of the mentioned

65. What is the output of this program?

```
interface calculate {  
    void cal(int item);  
}  
class displayA implements calculate {  
    int x;  
    public void cal(int item) {  
        x = item * item;  
    }  
}  
class displayB implements calculate {  
    int x;  
    public void cal(int item) {  
        x = item / item;  
    }  
}  
class interfaces {  
    public static void main(String args[]) {  
        displayA arr1 = new displayA;  
        displayB arr2 = new displayB;  
        arr1.x = 0;  
        arr2.x = 0;  
        arr1.cal(2);  
        arr2.cal(2);  
        System.out.print(arr1.x + " " + arr2.x);  
    }  
}
```

```
}  
}
```

- a) 0 0
- b) 2 2
- c) 4 1
- d) 1 4

66. What is the output of this program?

```
interface calculate {  
    int VAR = 0;  
    void cal(int item);  
}  
class display implements calculate {  
    int x;  
    public void cal(int item) {  
        if (item<2)  
            x = VAR;  
        else  
            x = item * item;  
    }  
}  
class interfaces {  
  
    public static void main(String args[]) {  
        display[] arr=new display[3];  
  
        for(int i=0;i<3;i++)  
            arr[i]=new display();  
        arr[0].cal(0);  
        arr[1].cal(1);  
        arr[2].cal(2);
```

```
        System.out.print(arr[0].x+" " + arr[1].x + " " + arr[2].x);  
    }  
}
```

- a) 0 1 2
- b) 0 2 4
- c) 0 0 4
- d) 0 1 4

67. Which of these is the method which is executed first before execution of any other thing takes place in a program?

- a) main method
- b) finalize method
- c) static method
- d) private method

68. What is the process of defining more than one method in a class differentiated by parameters?

- a) Function overriding
- b) Function overloading
- c) Function doubling
- d) None of these

69. Which of these can be used to differentiate two or more methods having same name?

- a) Parameters data type
- b) Number of parameters
- c) Return type of method
- d) All of the mentioned

70. Which of these data tupe can be used for a method having a return statement in it?

- a) void
- b) int
- c) float
- d) All of the mentioned.

71. Which of these statement is incorrect?

- a) Two or more methods with same name can be differentiated on the basis of their parameters data type.
- b) Two or more method having same name can be differentiated on basis of number of parameters.
- c) Any already defined method in java's library can be defined again in the program with different data type of parameters.
- d) If a method is returning a value the calling statement must have a variable to store that value.

72. What is the output of this program?

```
class box {  
    int width;  
    int height;  
    int length;  
    int volume;  
    void volume(int height, int length, int width) {  
        volume = width * height * length;  
    }  
}  
  
class Parameterized_method{  
    public static void main(String args[]) {  
        box obj = new box();  
        obj.height = 1;  
        obj.length = 5;  
        obj.width = 5;  
        obj.volume(3, 2, 1);  
    }  
}
```

```
        System.out.println(obj.volume);
    }
}
```

- a) 0
- b) 1
- c) 6
- d) 25

73. What is the output of this program?

```
class equality {
    int x;
    int y;
    boolean isequal(){
        return(x == y);
    }
}

class Output {
    public static void main(String args[]) {
        equality obj = new equality();
        obj.x = 5;
        obj.y = 5;
        System.out.println(obj.isequal);
    }
}
```

- a) false
- b) true
- c) 0
- d) 1

74. What is the output of this program?

```
class box {
```

```

int width;
int height;
int length;
int volume;
void volume() {
    volume = width * height * length;
}
void volume(int x) {
    volume = x;
}
}
class Output {
    public static void main(String args[]) {
        box obj = new box();
        obj.height = 1;
        obj.length = 5;
        obj.width = 5;
        obj.volume(5);
        System.out.println(obj.volume);
    }
}

```

- a) 0
- b) 5
- c) 25
- d) 26

75. What is the output of this program?

advertisements

```

class Output {
    static void main(String args[])
    {

```

```
int x , y = 1;
x = 10;
if(x != 10 && x / 0 == 0)
    System.out.println(y);
else
    System.out.println(++y);
}
```

- a) 1
- b) 2
- c) Runtime Error
- d) Compilation Error

76. What is the output of this program?

```
class area {
    int width;
    int length;
    int volume;
    area() {
        width = 5;
        length = 6;
    }
    void volume() {
        volume = width * height * length;
    }
}

class cons_method {
    public static void main(String args[]) {
        area obj = new area();
        obj.volume();
        System.out.println(obj.volume);
    }
}
```

```
}  
}
```

- a) 0
- b) 1
- c) 25
- d) 30

77. Which of these keywords are used to define an abstract class?

- a) abst b) abstract
- c) Abstract
- d) abstract class

78. Which of these is not abstract?

- a) Thread
- b) AbstractList
- c) List
- d) None of the Mentioned

79. If a class inheriting an abstract class does not define all of its function then it will be known as?

- a) abstract
- b) A simple class
- c) Static class
- d) None of the mentioned

80. Which of these is not a correct statement?

- a) Every class containing abstract method must be declared abstract.
- b) Abstract class defines only the structure of the class not its implementation.
- c) Abstract class can be initiated by new operator.
- d) Abstract class can be inherited.

81. Which of these packages contains abstract keyword?

- a) java.lang

- b) java.util
- c) java.io
- d) java.system

82. What is the output of this program?

```
class A {  
    public int i;  
    private int j;  
}  
class B extends A {  
    void display() {  
        super.j = super.i + 1;  
        System.out.println(super.i + " " + super.j);  
    }  
}  
class inheritance {  
    public static void main(String args[])  
    {  
        B obj = new B();  
        obj.i=1;  
        obj.j=2;  
        obj.display();  
    }  
}
```

- a) 2 2
- b) 3 3
- c) Runtime Error
- d) Compilation Error

83. What is the output of this program?

```
class A {
```

```

    public int i;
    public int j;
    A() {
        i = 1;
        j = 2;
    }
}
class B extends A {
    int a;
    B() {
        super();
    }
}
class super_use {
    public static void main(String args[])
    {
        B obj = new B();
        System.out.println(obj.i + " " + obj.j)
    }
}

```

- a) 1 2
- b) 2 1
- c) Runtime Error
- d) Compilation Error

84. What is the output of this program?

```

abstract class A {
    int i;
    abstract void display();
}
class B extends A {

```

```

    int j;
    void display() {
        System.out.println(j);
    }
}

class Abstract_demo {
    public static void main(String args[])
    {
        B obj = new B();
        obj.j=2;
        obj.display();
    }
}

```

- a) 0
- b) 2
- c) Runtime Error
- d) Compilation Error

85. What is the output of this program?

```

class A {
    int i;
    void display() {
        System.out.println(i);
    }
}

class B extends A {
    int j;
    void display() {
        System.out.println(j);
    }
}

```

```
class method_overriding {  
    public static void main(String args[])  
    {  
        B obj = new B();  
        obj.i=1;  
        obj.j=2;  
        obj.display();  
    }  
}
```

- a) 0
- b) 1
- c) 2
- d) Compilation Error

86. What is the output of this program?

```
class A {  
    public int i;  
    protected int j;  
}  
class B extends A {  
    int j;  
    void display() {  
        super.j = 3;  
        System.out.println(i + " " + j);  
    }  
}  
class Output {  
    public static void main(String args[])  
    {  
        B obj = new B();  
        obj.i=1;
```

```
        obj.j=2;
        obj.display();
    }
}
```

- a) 1 2
- b) 2 1
- c) 1 3
- d) 3 1

### UNIT-3

1. You can use the \_\_\_\_\_ method to force one thread to wait for another thread to finish. [ ]
  - a. sleep(long milliseconds) b. yield() c. stop() d. join()
2. A Java exception is an instance of \_\_\_\_\_. [ ]
  - a. Runtime Exception b. Exception c. Error d. Throw able
3. Program statements that are to be monitored for exceptions are contained within \_\_\_\_\_ block. [ ]
  - a. try b. catch c. throw d. throws
4. \_\_\_\_\_ modifier can be accessed only to classes in the same package.
5. \_\_\_\_\_ modifiers applies to variables only and it is not stored as part of its objects persistent state.

6. \_\_\_\_\_ package has the ability to analyze itself at runtime.

7.A \_\_\_\_\_ is a collection of classes and interfaces that provides a high-level layer of access protection and name space management.

8.Interfaces with no methods are called as \_\_\_\_\_

9.An instance of \_\_\_\_\_ are unchecked exceptions

10.An application responds to an exception by throwing an another exception is called as \_\_\_\_\_.

11.Benefits of exception handling

- a. Separating Error-Handling code from “regular” business logic code
- b. Propagating errors up the call stack
- c. Grouping and differentiating error types
- d. none

12. All exceptions are sub-classes of the build-in class

- a. Throwable b. serializable c. exception d. none

13. \_\_\_\_\_ defined automatically for user programs to include: division by zero, invalid array indexing, etc.

14. \_\_\_\_\_ is the most general and represents any type of error that can occur when performing I/O

15. Which of the following statement is not defined in the Object class?

- a) sleep(long milliseconds)      b) wait()      c) notify()      d) notifyAll()

16. The keyword to synchronize methods in Java is \_\_\_\_\_ [ ]

- a. synchronize b. synchronizing c. synchronized d. Synchronized

17. \_\_\_\_\_ condition occurs when two threads have a circular dependency on a pair of synchronized objects.

18 The wait(), notify(), and notify All() methods are implemented as \_\_\_\_\_ methods in Object class.

19. Sleep(10000) causes the main thread to sleep for seconds \_\_\_\_\_

20. Which method on a condition should you invoke to wake all waiting thread?

- a) condition.wake();
- b) condition.signal();
- c) condition.wakeAll();
- d) condition.signalAll();

21. You can use the method to force one thread to wait for another thread

- a) sleep(long milliseconds)
- b) yield()
- c) stop()
- d) join()

22. Which cannot directly cause a thread to stop executing?

- a) Calling a yield method
- b) Calling the start method on another thread object
- c) Calling the notify method on object
- d) Calling wait method on an object

23. With \_\_\_\_\_ many threads share the same object instance.

24. A thread is a

- A) light
- B) strong
- C) small
- D) large

25. This thread is called the \_\_\_\_\_ thread because it is the thread that executes when you start the main program.

- A) float
- B) main
- C) void
- D) int

26. When does Exceptions in Java arises in code sequence?

- a) Run Time
- b) Compilation Time
- c) Can Occur Any Time
- d) None of the mentioned

27. Which of these keywords is not a part of exception handling?

- a) try
- b) finally
- c) thrown
- d) catch

28. Which of these keywords must be used to monitor for exceptions?

a) try b) finally c) throw d) catch

29. Which of these keywords must be used to handle the exception thrown by try block in some rational manner?

a) try b) finally c) throw d) catch

30. Which of these keywords is used to manually throw an exception?

a) try b) finally c) throw d) catch

31. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  System.out.print("Hello" + " " + 1 / 0);
5.              }
6.          catch(ArithmeticException e) {
7.              System.out.print("World");
8.          }
9.      }
10.     }
```

a) Hello b) World c) HelloWorld d) Hello World

[View Answer](#)

31. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  int a, b;
5.                  b = 0;
6.                  a = 5 / b;
7.                  System.out.print("A");
```

```
8.         }
9.         catch(ArithmeticException e) {
10.            System.out.print("B");
11.        }
12.    }
13. }
```

a) A b) B c) Compilation Error d) Runtime Error

32. What is the output of this program?

```
1.     class exception_handling {
2.         public static void main(String args[]) {
3.             try {
4.                 int a, b;
5.                 b = 0;
6.                 a = 5 / b;
7.                 System.out.print("A");
8.             }
9.             catch(ArithmeticException e) {
10.                System.out.print("B");
11.            }
12.            finally {
13.                System.out.print("C");
14.            }
15.        }
16.    }
```

a) A b) B c) AC d) BC

33. What is the output of this program?

```
1.     class exception_handling {
2.         public static void main(String args[]) {
3.             try {
4.                 int i, sum;
```

```

5.         sum = 10;
6.         for (i = -1; i < 3 ;++i)
7.             sum = (sum / i);
8.         }
9.         catch(ArithmeticException e) {
10.            System.out.print("0");
11.        }
12.        System.out.print(sum);
13.    }
14.    }

```

a) 0 b) 05 c) Compilation Error d) Runtime Error

34. What is the output of this program?

```

1.     class exception_handling {
2.         public static void main(String args[]) {
3.             try {
4.                 int i, sum;
5.                 sum = 10;
6.                 for (i = -1; i < 3 ;++i) {
7.                     sum = (sum / i);
8.                     System.out.print(i);
9.                 }
10.            }
11.            catch(ArithmeticException e) {
12.                System.out.print("0");
13.            }
14.        }
15.    }

```

a) -1 b) 0 c) -10 d) -101

35. Which of these is a super class of all exceptional type classes?

- a) String
- b) RuntimeExceptions
- c) Throwable
- d) Cachable

36. Which of these class is related to all the exceptions that can be caught by using catch?

- a) Error
- b) Exception
- c) RuntimeException
- d) All of the mentioned

37. Which of these class is related to all the exceptions that cannot be caught?

- a) Error
- b) Exception
- c) RuntimeException
- d) All of the mentioned

38. Which of these handles the exception when no catch is used?

- a) Default handler
- b) finally
- c) throw handler
- d) Java run time system

39. Which of these keywords is used to manually throw an exception?

- a) try
- b) finally
- c) throw
- d) catch

40. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  System.out.print("Hello" + " " + 1 / 0);
5.              }
6.              finally {
7.                  System.out.print("World");
8.              }
9.          }
10.     }
```

- a) Hello
- b) World
- c) Compilation Error
- d) First Exception then World

41. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  int a, b;
5.                  b = 0;
6.                  a = 5 / b;
7.                  System.out.print("A");
8.              }
9.              catch(ArithmeticException e) {
10.                 System.out.print("B");
11.             }
12.         }
13.     }
```

- a) A
- b) B

- c) Compilation Error
- d) Runtime Error

42. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  int a[] = {1, 2,3 , 4, 5};
5.                  for (int i = 0; i < 7; ++i)
6.                      System.out.print(a[i]);
7.              }
8.          catch(ArrayIndexOutOfBoundsException e) {
9.              System.out.print("0");
10.         }
11.     }
12. }
```

- a) 12345
- b) 123450
- c) 1234500
- d) Compilation Error

43. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  int i, sum;
5.                  sum = 10;
6.                  for (i = -1; i < 3 ;++i)
7.                      sum = (sum / i);
8.              }
9.          catch(ArithmeticException e) {
```

```
10.         System.out.print("0");
11.         }
12.         System.out.print(sum);
13.         }
14.         }
```

- a) 0
- b) 05
- c) Compilation Error
- d) Runtime Error

44. What is the output of this program?

```
1.         class exception_handling {
2.             public static void main(String args[]) {
3.                 try {
4.                     int a[] = {1, 2,3 , 4, 5};
5.                     for (int i = 0; i < 5; ++i)
6.                         System.out.print(a[i]);
7.                     int x = 1/0;
8.                 }
9.                 catch(ArrayIndexOutOfBoundsException e) {
10.                    System.out.print("A");
11.                }
12.                catch(ArithmeticException e) {
13.                    System.out.print("B");
14.                }
15.            }
16.        }
```

- a) 12345
- b) 12345A
- c) 12345B

d) Comiplation Error

45. Which of these keywords is used to generate an exception explicitly?

- a) try
- b) finally
- c) throw
- d) catch

46. Which of these class is related to all the exceptions that are explicitly thrown?

- a) Error
- b) Exception
- c) Throwable
- d) Throw

47. Which of these operator is used to generate an instance of an exception than can be thrown by using throw?

- a) new
- b) malloc
- c) alloc
- d) thrown

48 Which of these handles the exception when no catch is used?

- a) Default handler
- b) finally
- c) throw handler
- d) Java run time system

49. Which of these keywords is used to by the calling function to guard against the exception that is thrown by called function?

- a) try
- b) throw

c) throws

d) catch

50. What is the output of this program?

```
1.      class exception_handling {  
2.          public static void main(String args[]) {  
3.              try {  
4.                  int a = args.length;  
5.                  int b = 10 / a;  
6.                  System.out.print(a);  
7.                  try {  
8.                      if (a == 1)  
9.                          a = a / a - a;  
10.                     if (a == 2) {  
11.                         int c = {1};  
12.                         c[8] = 9;  
13.                     }  
14.                 }  
15.                 catch (ArrayIndexOutOfBoundsException e) {  
16.                     System.out.println("TypeA");  
17.                 }  
18.                 catch (ArithmeticException e) {  
19.                     System.out.println("TypeB");  
20.                 }  
21.             }  
22.         }  
23.     }
```

a) TypeA

b) TypeB

c) 0TypeA

d) 0TypeB

51. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  throw new NullPointerException ("Hello");
5.                  System.out.print("A");
6.              }
7.          catch(ArithmeticException e) {
8.              System.out.print("B");
9.          }
10.     }
11. }
```

- a) A
- b) B
- c) Hello
- d) Runtime Error

52. What is the output of this program?

```
1.      class exception_handling {
2.          static void throwexception() throws ArithmeticException {
3.              System.out.print("0");
4.              throw new ArithmeticException ("Exception");
5.          }
6.          public static void main(String args[]) {
7.              try {
8.                  throwexception();
9.              }
10.             catch (ArithmeticException e) {
11.                 System.out.println("A");

```

```
12.         }  
13.         }  
14.         }
```

- a) A
- b) 0
- c) 0A
- d) Exception

53. What is the output of this program?

```
1.     class exception_handling {  
2.         public static void main(String args[]) {  
3.             try {  
4.                 int a = args.length;  
5.                 int b = 10 / a;  
6.                 System.out.print(a);  
7.                 try {  
8.                     if (a == 1)  
9.                         a = a / a - a;  
10.                    if (a == 2) {  
11.                        int c = {1};  
12.                        c[8] = 9;  
13.                    }  
14.                }  
15.                catch (ArrayIndexOutOfBoundsException e) {  
16.                    System.out.println("TypeA");  
17.                }  
18.                catch (ArithmeticException e) {  
19.                    System.out.println("TypeB");  
20.                }  
21.            }  
22.        }
```

23.            }

- a) TypeA
- b) TypeB
- c) 0TypeA
- d) 0TypeB

54. What is the output of this program?

```
1.            class exception_handling {
2.                public static void main(String args[]) {
3.                    try {
4.                        int a = args.length;
5.                        int b = 10 / a;
6.                        System.out.print(a);
7.                        try {
8.                            if (a == 1)
9.                                a = a / a - a;
10.                            if (a == 2) {
11.                                int c = {1};
12.                                c[8] = 9;
13.                            }
14.                        }
15.                        catch (ArrayIndexOutOfBoundsException e) {
16.                            System.out.println("TypeA");
17.                        }
18.                        catch (ArithmeticException e) {
19.                            System.out.println("TypeB");
20.                        }
21.                    }
22.                }
23.            }
```

- a) TypeA
- b) TypeB
- c) 0TypeA
- d) 0TypeB

55. Which of these method can be used to make the main thread to be executed last among all the threads?

- a) stop()
- b) sleep()
- c) join()
- d) call()

56. Which of these method is used to find out that a thread is still running or not?

- a) run()
- b) Alive()
- c) isAlive()
- d) checkRun()

57. What is the default value of priority variable MIN\_PRIORITY AND MAX\_PRIORITY?

- a) 0 & 256
- b) 0 & 1
- c) 1 & 10
- d) 1 & 256

58. Which of these method waits for the thread to terminate?

- a) sleep()
- b) isAlive()
- c) join()
- d) stop()

59. Which of these method is used to explicitly set the priority of a thread?

- a) set()
- b) make()
- c) setPriority()
- d) makePriority()

60. What is synchronization in reference to a thread?

- a) It's a process of handling situations when two or more threads need access to a shared resource.
- b) Its a process by which many thread are able to access same shared resource simultaneously.
- c) Its a process by which a method is able to access many different threads simultaneously.
- d) Its a method that allow to many threads to access any information require.

61. What is the output of this program?

```
1.      class newthread extends Thread {  
2.          newthread() {  
3.              super("My Thread");  
4.              start();  
5.          }  
6.          public void run() {  
7.              System.out.println(this);  
8.          }  
9.      }  
10.     class multithreaded_programing {  
11.         public static void main(String args[]) {  
12.             new newthread();  
13.         }  
14.     }
```

- a) My Thread
- b) Thread[My Thread,5,main] c) Compilation Error

d) Runtime Error

62. What is the output of this program?

```
1.      class newthread extends Thread {  
2.          Thread t;  
3.          newthread() {  
4.              t = new Thread(this, "My Thread");  
5.              t.start();  
6.          }  
7.          public void run() {  
8.              try {  
9.                  t.join()  
10.                 System.out.println(t.getName());  
11.             }  
12.             catch(Exception e) {  
13.                 System.out.print("Exception");  
14.             }  
15.         }  
16.     }  
17.     class multithreaded_programing {  
18.         public static void main(String args[]) {  
19.             new newthread();  
20.         }  
21.     }
```

- a) My Thread
- b) Thread[My Thread,5,main] c) Exception
- d) Runtime Error

63. What is the output of this program?

```
1.      class newthread extends Thread {  
2.          Thread t;
```

```

3.         newthread() {
4.             t = new Thread(this,"New Thread");
5.             t.start();
6.         }
7.         public void run() {
8.             System.out.println(t.isAlive());
9.         }
10.    }
11.    class multithreaded_programing {
12.        public static void main(String args[]) {
13.            new newthread();
14.        }
15.    }

```

- a) 0
- b) 1
- c) true
- d) false

64. What is the output of this program?

```

1.         class newthread extends Thread {
2.             Thread t1,t2;
3.             newthread() {
4.                 t1 = new Thread(this,"Thread_1");
5.                 t2 = new Thread(this,"Thread_2");
6.                 t1.start();
7.                 t2.start();
8.             }
9.             public void run() {
10.                t2.setPriority(Thread.MAX_PRIORITY);
11.                System.out.print(t1.equals(t2));
12.            }

```

```
13.     }
14.     class multithreaded_programing {
15.         public static void main(String args[]) {
16.             new newthread();
17.         }
18.     }
```

- a) true
- b) false
- c) true>true
- d) false>false

65. Which of these method is used to implement Runnable interface?

- a) stop()
- b) run()
- c) runThread()
- d) stopThread()

66. Which of these interface is implemented by Thread class?

- a) Runnable
- b) Connections
- c) Set
- d) MapConnections

67. Which of these method is used to begin the execution of a thread?

- a) run()
- b) start()
- c) runThread()
- d) startThread()

68. Which of these method waits for the thread to terminate?

- a) sleep()

- b) isAlive()
- c) join()
- d) stop()

69. Which of these statement is incorrect?

- a) A thread can be formed by implementing Runnable interface only.
- b) A thread can be formed by a class that extends Thread class.
- c) start() method is used to begin execution of the thread.
- d) run() method is used to begin execution of a thread before start() method in special cases.

70. What is the output of this program?

```
1.      class newthread implements Runnable {  
2.          Thread t;  
3.          newthread() {  
4.              t = new Thread(this, "My Thread");  
5.              t.start();  
6.          }  
7.          public void run() {  
8.              System.out.println(t.getName());  
9.          }  
10.     }  
11.     class multithreaded_programing {  
12.         public static void main(String args[]) {  
13.             new newthread();  
14.         }  
15.     }
```

- a) My Thread
- b) Thread[My Thread,5,main] c) Compilation Error
- d) Runtime Error

71. What is the output of this program?

```

1.      class newthread implements Runnable {
2.          Thread t;
3.          newthread() {
4.              t = new Thread(this,"My Thread");
5.              t.start();
6.          }
7.          public void run() {
8.              System.out.println(t);
9.          }
10.     }
11.     class multithreaded_programing {
12.         public static void main(String args[]) {
13.             new newthread();
14.         }
15.     }

```

- a) My Thread
- b) Thread[My Thread,5,main] c) Compilation Error
- d) Runtime Error

72. What is the output of this program?

```

1.      class newthread implements Runnable {
2.          Thread t;
3.          newthread() {
4.              t = new Thread(this,"My Thread");
5.              t.start();
6.          }
7.     }
8.     class multithreaded_programing {
9.         public static void main(String args[]) {
10.            new newthread();
11.        }

```

12.            }

- a) My Thread
- b) Thread[My Thread,5,main] c) Compilation Error
- d) Runtime Error

73. What is the output of this program?

```
1.      class newthread implements Runnable {
2.          Thread t;
3.          newthread() {
4.              t = new Thread(this,"New Thread");
5.              t.start();
6.          }
7.          public void run() {
8.              t.setPriority(Thread.MAX_PRIORITY);
9.              System.out.println(t);
10.         }
11.     }
12.     class multithreaded_programing {
13.         public static void main(String args[]) {
14.             new newthread();
15.         }
16.     }
```

- a) Thread[New Thread,0,main] b) Thread[New Thread,1,main] c) Thread[New Thread,5,main]
- d) Thread[New Thread,10,main] ]

74. What is the output of this program?

```
1.      class newthread implements Runnable {
2.          Thread t;
3.          newthread() {
4.              t1 = new Thread(this,"Thread_1");
5.              t2 = new Thread(this,"Thread_2");
6.              t1.start();
```

```

7.         t2.start();
8.         }
9.         public void run() {
10.            t2.setPriority(Thread.MAX_PRIORITY);
11.            System.out.print(t1.equals(t2));
12.        }
13.    }
14.    class multithreaded_programing {
15.        public static void main(String args[]) {
16.            new newthread();
17.        }
18.    }

```

- a) true
- b) false
- c) true true
- d) false false

75. Which of these class is used to make a thread?

- a) String
- b) System
- c) Thread
- d) Runnable

76. Which of these interface is implemented by Thread class?

- a) Runnable
- b) Connections
- c) Set
- d) MapConnections

77. Which of these method of Thread class is used to find out the priority given to a thread?

- a) get()

- b) ThreadPriority()
- c) getPriority()
- d) getThreadPriority()

78. Which of these method of Thread class is used to Suspend a thread for a period of time?

- a) sleep()
- b) terminate()
- c) suspend()
- d) stop()

79. Which function of pre defined class Thread is used to check weather current thread being checked is still running?

- a) isAlive()
- b) Join()
- c) isRunning()
- d) Alive()

80. What is the output of this program?

```
1.      class multithreaded_programing {  
2.          public static void main(String args[]) {  
3.              Thread t = Thread.currentThread();  
4.              t.setName("New Thread");  
5.              System.out.println(t);  
6.          }  
7.      }
```

- a) Thread[5,main] b) Thread[New Thread,5] c) Thread[main,5,main] d) Thread[New Thread,5,main]

81. What is the priority of the thread in output of this program?

```
1.      class multithreaded_programing {  
2.          public static void main(String args[]) {  
3.              Thread t = Thread.currentThread();
```

```
4.         t.setName("New Thread");
5.         System.out.println(t.getName());
6.     }
7. }
```

- a) main
- b) Thread
- c) New Thread
- d) Thread[New Thread,5,main]

82. What is the name of the thread in output of this program?

```
1.     class multithreaded_programing {
2.         public static void main(String args[]) {
3.             Thread t = Thread.currentThread();
4.             System.out.println(t.getPriority());
5.         }
6.     }
```

- a) 0
- b) 1
- c) 4
- d) 5

83. What is the name of the thread in output of this program?

```
1.     class multithreaded_programing {
2.         public static void main(String args[]) {
3.             Thread t = Thread.currentThread();
4.             System.out.println(t.isAlive());
5.         }
6.     }
```

- a) 0
- b) 1
- c) true

d) false

84. What is multithreaded programming?

- a) It's a process in which two different processes run simultaneously.
- b) It's a process in which two or more parts of same process run simultaneously.
- c) Its a process in which many different process are able to access same information.
- d) Its a process in which a single process can access information from many sources.

85 Which of these are types of multitasking?

- a) Process based
- b) Thread based
- c) Process and Thread based
- d) None of the mentioned

86. Which of these packages contain all the Java's built in exceptions?

- a) java.io
- b) java.util
- c) java.lang
- d) java.net

87. Thread priority in Java is?

- a) Integer
- b) Float
- c) double
- d) long

88. What will happen if two thread of same priority are called to be processed simultaneously?

- a) Any one will be executed first lexographically
- b) Both of them will be executed simultaneously
- c) None of them will be executed

d) It is dependent on the operating system.

89. Which of these statements is incorrect?

- a) By multithreading CPU's idle time is minimized, and we can take maximum use of it.
- b) By multitasking CPU's idle time is minimized, and we can take maximum use of it.
- c) Two thread in Java can have same priority
- d) A thread can exist only in two states, running and blocked.

90. What is the output of this program?

```
1.      class multithreaded_programing {  
2.          public static void main(String args[]) {  
3.              Thread t = Thread.currentThread();  
4.              System.out.println(t);  
5.          }  
6.      }
```

a) Thread[5,main] b) Thread[main,5] c) Thread[main,0] d) Thread[main,5,main]View Answer

91. What is the priority of the thread in output of this program?

```
1.      class multithreaded_programing {  
2.          public static void main(String args[]) {  
3.              Thread t = Thread.currentThread();  
4.              System.out.println(t);  
5.          }  
6.      }
```

- a) 4
- b) 5
- c) 0
- d) 1

92. What is the name of the thread in output of this program?

```
1.      class multithreaded_programing {  
2.          public static void main(String args[]) {
```

```
3.         Thread t = Thread.currentThread();
4.         System.out.println(t);
5.     }
6. }
```

- a) main
- b) Thread
- c) System
- d) None of the mentioned

93. What is the name of the thread in output of this program?

```
1.     class multithreaded_programing {
2.         public static void main(String args[]) {
3.             Thread t = Thread.currentThread();
4.             System.out.println(t.isAlive());
5.         }
6.     }
```

- a) 0
- b) 1
- c) true
- d) false

94. Which of these clause will be executed even if no exceptions are found?

- a) throws
- b) finally
- c) throw
- d) catch

94. A single try block must be followed by which of these?

- a) finally
- b) catch
- c) finally & catch

d) None of the mentioned

95. Which of these packages contain all the Java's built in exceptions?

a) java.io

b) java.util

c) java.lang

d) java.net

96. Which of these exceptions handles the divide by zero error?

a) ArithmeticException

b) MathException

c) IllegalAccessException

d) IllegalException

97. Which of these exceptions will occur if we try to access the index of an array beyond its length?

a) ArithmeticException

b) ArrayException

c) ArrayIndexException

d) ArrayIndexOutOfBoundsException

98. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  int a = args.length;
5.                  int b = 10 / a;
6.                  System.out.print(a);
7.              }
8.          catch (ArithmeticException e) {
9.              System.out.println("1");
10.         }
11.     }
12. }
```

- a) 0
- b) 1
- c) Compilation Error
- d) Runtime Error

Note : Execution command line : \$ java exception\_handling

99. What is the output of this program?

```
1.      class exception_handling {
2.          public static void main(String args[]) {
3.              try {
4.                  throw new NullPointerException ("Hello");
5.                  System.out.print("A");
6.              }
7.          catch(ArithmeticException e) {
8.              System.out.print("B");
9.          }
10.     }
11. }
```

- a) A
- b) B
- c) Compilation Error
- d) Runtime Error

100. What is the output of this program?

```
1.      class exception_handling {
2.          static void throwexception() throws ArithmeticException {
3.              System.out.print("0");
4.              throw new ArithmeticException ("Exception");
5.          }
6.          public static void main(String args[]) {
7.              try {
8.                  throwexception();
9.              }
```

```
10.         catch (ArithmeticException e) {
11.             System.out.println("A");
12.         }
13.     }
14. }
```

- a) A
- b) 0
- c) 0A
- d) Exception

101. What is the output of this program?

```
1.     class exception_handling
2.     {
3.         public static void main(String args[])
4.         {
5.             try
6.             {
7.                 int a = 1;
8.                 int b = 10 / a;
9.                 try
10.                {
11.                    if (a == 1)
12.                        a = a / a - a;
13.                    if (a == 2)
14.                        {
15.                            int c[] = {1};
16.                            c[8] = 9;
17.                        }
18.                }
19.            finally
20.            {
```

```
21.         System.out.print("A");
22.     }
23. }
24.     catch (Exception e)
25.     {
26.         System.out.println("B");
27.     }
28. }
29. }
```

- a) A
- b) B
- c) AB
- d) BA

102. What is the output of this program?

```
1.     class exception_handling {
2.         public static void main(String args[]) {
3.             try {
4.                 int a = args.length;
5.                 int b = 10 / a;
6.                 System.out.print(a);
7.                 try {
8.                     if (a == 1)
9.                         a = a / a - a;
10.                    if (a == 2) {
11.                        int c = {1};
12.                        c[8] = 9;
13.                    }
14.                }
15.            } catch (ArrayIndexOutOfBoundsException e) {
16.                System.out.println("TypeA");
17.            }
```

```
18.         catch (ArithmeticException e) {
19.             System.out.println("TypeB");
20.         }
21.     }
22. }
```

- a) TypeA
- b) TypeB
- c) Compilation Error
- d) Runtime Error

103. Which of these keywords are used to implement synchronization?

- a) sunchronize
- b) syn
- c) synch
- d) synchronized

104. Which of these method is used to avoid polling in Java?

- a) wait()
- b) notify()
- c) notifyAll()
- d) All of the mentioned

105. Which of these method is used to tell the calling thread to give up monitor and go to sleep until some other thread enters the same monitor?

- a) wait()
- b) notify()
- c) notifyAll()
- d) sleep()

106. Which of these method wakes up the first thread that called wait()?

- a) wake()
- b) notify()
- c) start()
- d) notifyAll()

107. Which of these method wakes up all the threads?

- a) wakeAll()
- b) notify()
- c) start()
- d) notifyAll()

108. What is synchronization in reference to a thread?

- a) Its a process of handling situations when two or more threads need access to a shared resource.
- b) Its a process by which many thread are able to access same shared resource simultaneously.
- c) Its a process by which a method is able to access many different threads simultaneously.
- d) Its a method that allow to many threads to access any information the require.

109. What is the output of this program?

```
1.      class newthread extends Thread {  
2.          Thread t;  
3.          String name;  
4.          newthread(String threadname) {  
5.              name = threadname;  
6.              t = new Thread(this,name);  
7.              t.start();  
8.          }  
9.          public void run() {  
10.         }  
11.  
12.     }
```

```

13.      class multithreaded_programing {
14.          public static void main(String args[]) {
15.              newthread obj1 =      new newthread("one");
16.              newthread obj2 =      new newthread("two");
17.          try {
18.              obj1.t.wait();
19.              System.out.print(obj1.t.isAlive());
20.          }
21.          catch(Exception e) {
22.              System.out.print("Main thread interrupted");
23.          }
24.      }
25.  }

```

- a) true
- b) false
- c) Main thread interrupted
- d) None of the mentioned

110. What is the output of this program?

```

1.      class newthread extends Thread {
2.          Thread t;
3.          String name;
4.          newthread(String threadname) {
5.              name = threadname;
6.              t = new Thread(this,name);
7.              t.start();
8.          }
9.          public void run() {
10.         }
11.
12.     }

```

```

13.      class multithreaded_programing {
14.          public static void main(String args[]) {
15.              newthread obj1 =      new newthread("one");
16.              newthread obj2 =      new newthread("two");
17.          try {
18.              Thread.sleep(1000);
19.              System.out.print(obj1.t.isAlive());
20.          }
21.          catch(InterruptedException e) {
22.              System.out.print("Main thread interrupted");
23.          }
24.      }
25.  }

```

- a) true
- b) false
- c) Main thread interrupted
- d) None of the mentioned

111. What is the output of this program?

```

1.      class newthread extends Thread {
2.          Thread t;
3.          String name;
4.          newthread(String threadname) {
5.              name = threadname;
6.              t = new Thread(this,name);
7.              t.start();
8.          }
9.          public void run() {
10.         }
11.
12.     }

```

```

13.      class multithreaded_programing {
14.          public static void main(String args[]) {
15.              newthread obj1 =      new newthread("one");
16.              newthread obj2 =      new newthread("two");
17.          try {
18.              System.out.print(obj1.t.equals(obj2.t));
19.          }
20.          catch(Exception e) {
21.              System.out.print("Main thread interrupted");
22.          }
23.      }
24.  }

```

- a) true
- b) false
- c) Main thread interrupted
- d) None of the mentioned

112. What is the output of this program?

```

1.      class newthread extends Thread {
2.          Thread t;
3.          newthread() {
4.              t1 = new Thread(this, "Thread_1");
5.              t2 = new Thread(this, "Thread_2");
6.              t1.start();
7.              t2.start();
8.          }
9.          public void run() {
10.             t2.setPriority(Thread.MAX_PRIORITY);
11.             System.out.print(t1.equals(t2));
12.         }
13.     }

```

```

14.      class multithreaded_programing {
15.          public static void main(String args[]) {
16.              new newthread();
17.          }
18.      }

```

- a) true
- b) false
- c) truetrue
- d) falsefalse

.

#### Unit-4

1. Which of these packages contain all the collection classes?

- a) java.lang b) java.util c) java.net d) java.awt

2. Which of these classes is not part of Java's collection framework?

- a) Maps b) Array c) Stack d) Queue

3. Which of these interface is not a part of Java's collection framework?

- a) List b) Set c) SortedMap d) SortedList

4. Which of these methods deletes all the elements from invoking collection?

- a) clear() b) reset() c) delete() d) refresh()

5. What is Collection in Java?

- a) A group of objects b) A group of classes c) A group of interfaces d) None of the mentioned

6. What is the output of this program?

```

1.      import java.util.*;
2.      class Array {
3.          public static void main(String args[]) {
4.              int array[] = new int [5];
5.              for (int i = 5; i > 0; i--)
6.                  array[5-i] = i;
7.              Arrays.fill(array, 1, 4, 8);

```

```
8.         for (int i = 0; i < 5 ; i++)
9.             System.out.print(array[i]);
10.        }
11.    }
```

a) 12885 b) 12845 c) 58881 d) 54881

7. What is the output of this program?

```
1.     import java.util.*;
2.     class vector {
3.         public static void main(String args[] ) {
4.             Vector obj = new Vector(4,2);
5.             obj.addElement(new Integer(3));
6.             obj.addElement(new Integer(2));
7.             obj.addElement(new Integer(5));
8.             obj.removeAll(obj);
9.             System.out.println(obj.isEmpty());
10.        }
11.    }
```

a) 0 b) 1 c) true d) false

8. What is the output of this program?

```
1.     import java.util.*;
2.     class stack {
3.         public static void main(String args[] ) {
4.             Stack obj = new Stack();
5.             obj.push(new Integer(3));
6.             obj.push(new Integer(2));
7.             obj.pop();
8.             obj.push(new Integer(5));
9.             System.out.println(obj);
10.        }
```

11.            }

a) [3, 5] b) [3, 2] c) [3, 2, 5] d) [3, 5, 2]

9. What is the output of this program?

1.            **import** java.util.\*;

2.            **class** hashtable {

3.            **public static void** main(String args[]) {

4.            Hashtable obj = **new** Hashtable();

5.            obj.put("A", **new** Integer(3));

6.            obj.put("B", **new** Integer(2));

7.            obj.put("C", **new** Integer(8));

8.            obj.remove(**new** String("A"));

9.            System.out.print(obj);

10.           }

11.           }

a) {C=8, B=2} b) [C=8, B=2] c) {A=3, C=8, B=2} d) [A=3, C=8, B=2] View Answer

10. What is the output of this program?

1.            **import** java.util.\*;

2.            **class** BitSet {

3.            **public static void** main(String args[]) {

4.            BitSet obj = **new** BitSet(5);

5.            **for** (int i = 0; i < 5; ++i)

6.            obj.set(i);

7.            obj.clear(2);

8.            System.out.print(obj);

9.            }

10.           }

a) {0, 1, 3, 4} b) {0, 1, 2, 4} c) {0, 1, 2, 3, 4} d) {0, 0, 0, 3, 4}

11. Which of these standard collection classes implements a dynamic array?

a) AbstractList b) LinkedList c) ArrayList d) AbstractSet

12. Which of these class can generate an array which can increase and decrease in size automatically?

a) ArrayList() b) DynamicList() c) LinkedList() d) DynamicList()

13. Which of these method can be used to increase the capacity of ArrayList object manually?

a) Capacity() b) increaseCapacity() c) increasecapacity() d) ensureCapacity()

14. Which of these method of ArrayList class is used to obtain present size of an object?

a) size() b) length() c) index() d) capacity()

15. Which of these methods can be used to obtain a static array from an ArrayList object?

a) Array() b) covertArray() c) toArray() d) coverttoArray()

16. Which of these method is used to reduce the capacity of an ArrayList object?

a) trim() b) trimSize() c) trimTosize() d) trimToSize()

17. What is the output of this program?

```
1.      import java.util.*;
2.      class Arraylist {
3.          public static void main(String args[]) {
4.              ArrayList obj = new ArrayList();
5.              obj.add("A");
6.              obj.add("B");
7.              obj.add("C");
8.              obj.add(1, "D");
9.              System.out.println(obj);
10.         }
11.     }
```

a) [A, B, C, D] b) [A, D, B, C] c) [A, D, C] d) [A, B, C]

18. What is the output of this program?

```
1.      import java.util.*;
2.      class Output {
3.          public static void main(String args[]) {
```

```
4.     ArrayList obj = new ArrayList();
5.     obj.add("A");
6.     obj.add(0, "B");
7.     System.out.println(obj.size());
8.     }
9.     }
```

a) 0 b) 1 c) 2 d) Any Garbage Value

19. What is the output of this program?

```
1.     import java.util.*;
2.     class Output {
3.         public static void main(String args[]) {
4.             ArrayList obj = new ArrayList();
5.             obj.add("A");
6.             obj.ensureCapacity(3);
7.             System.out.println(obj.size());
8.         }
9.     }
```

a) 1 b) 2 c) 3 d) 4

20. What is the output of this program?

```
1.     class Output {
2.         public static void main(String args[]) {
3.             ArrayList obj = new ArrayList();
4.             obj.add("A");
5.             obj.add("D");
6.             obj.ensureCapacity(3);
7.             obj.trimToSize();
8.             System.out.println(obj.size());
9.         }
10.    }
```

a) 1 b) 2 c) 3 d) 4

21. Which of these standard collection classes implements a linked list data structure?

a) AbstractList b) LinkedList c) HashSet d) AbstractSet

22. Which of these classes implements Set interface?

a) ArrayList b) HashSet c) LinkedList d) DynamicList

23. Which of these method is used to add an element to the start of a LinkedList object?

a) add() b) first() c) AddFirst() d) addFirst()

24. Which of these method of HashSet class is used to add elements to its object?

a) add() b) Add() c) addFirst() d) insert()

25. Which of these methods can be used to delete the last element in a LinkedList object?

a) remove() b) delete() c) removeLast() d) deleteLast()

26. Which of these method is used to change an element in a LinkedList Object?

a) change() b) set() c) redo() d) add()

27. What is the output of this program?

```
1.      import java.util.*;
2.      class LinkedList {
3.          public static void main(String args[]) {
4.              LinkedList obj = new LinkedList();
5.              obj.add("A");
6.              obj.add("B");
7.              obj.add("C");
8.              obj.addFirst("D");
9.              System.out.println(obj);
10.         }
11.     }
```

a) [A, B, C] b) [D, B, C] c) [A, B, C, D] d) [D, A, B, C]

28. What is the output of this program?

```
1.      import java.util.*;
2.      class LinkedList {
3.          public static void main(String args[]) {
```

```
4.         LinkedList obj = new LinkedList();
5.         obj.add("A");
6.         obj.add("B");
7.         obj.add("C");
8.         obj.removeFirst();
9.         System.out.println(obj);
10.        }
11.        }
```

a) [A, B] b) [B, C] c) [A, B, C, D] d) [A, B, C]

29. What is the output of this program?

```
1.         import java.util.*;
2.         class Output {
3.             public static void main(String args[] {
4.                 HashSet obj = new HashSet();
5.                 obj.add("A");
6.                 obj.add("B");
7.                 obj.add("C");
8.                 System.out.println(obj + " " + obj.size());
9.             }
10.        }
```

a) ABC 3 b) [A, B, C] 3 c) ABC 2 d) [A, B, C] 2

30. What is the output of this program?

```
1.         import java.util.*;
2.         class Output {
3.             public static void main(String args[] {
4.                 TreeSet t = new TreeSet();
5.                 t.add("3");
6.                 t.add("9");
7.                 t.add("1");
8.                 t.add("4");
9.                 t.add("8");
```

```
10.          System.out.println(t);
11.          }
12.          }
```

a) [1, 3, 5, 8, 9] b) [3, 4, 1, 8, 9] c) [9, 8, 4, 3, 1] d) [1, 3, 4, 8, 9]

31. Which of these object stores association between keys and values?

a) Hash table b) Map c) Array d) String

32. Which of these classes provide implementation of map interface?

a) ArrayList b) HashMap c) LinkedList d) DynamicList

33. Which of these method is used to remove all keys/values pair from the invoking map?

a) delete() b) remove() c) clear() d) removeAll()

34. Which of these method Map class is used to obtain an element in the map having specified key?

a) search() b) get() c) set() d) look()

35. Which of these methods can be used to obtain set of all keys in a map?

a) getAll() b) getKeys() c) keyall() d) keySet()

36. Which of these method is used add an element and corresponding key to a map?

a) put() b) set() c) redo() d) add()

37. What is the output of this program?

```
1.          import java.util.*;
2.          class Maps {
3.              public static void main(String args[]) {
4.                  HashMap obj = new HashMap();
5.                  obj.put("A", new Integer(1));
6.                  obj.put("B", new Integer(2));
7.                  obj.put("C", new Integer(3));
```

```
8.         System.out.println(obj);
9.         }
10.        }
```

a) {A 1, B 1, C 1} b) {A, B, C} c) {A-1, B-1, C-1} d) {A=1, B=2, C=3}

38. What is the output of this program?

```
1.         import java.util.*;
2.         class Maps {
3.             public static void main(String args[] {
4.                 HashMap obj = new HashMap();
5.                 obj.put("A", new Integer(1));
6.                 obj.put("B", new Integer(2));
7.                 obj.put("C", new Integer(3));
8.                 System.out.println(obj.keySet());
9.             }
10.        }
```

a) [A, B, C] b) {A, B, C} c) {1, 2, 3} d) [1, 2, 3] Answer: a

39. What is the output of this program?

```
1.         import java.util.*;
2.         class Maps {
3.             public static void main(String args[] {
4.                 HashMap obj = new HashMap();
5.                 obj.put("A", new Integer(1));
6.                 obj.put("B", new Integer(2));
7.                 obj.put("C", new Integer(3));
8.                 System.out.println(obj.get("B"));
9.             }
10.        }
```

a) 1 b) 2 c) 3 d) null

40. What is the output of this program?

```
1.      import java.util.*;
2.      class Maps {
3.          public static void main(String args[]) {
4.              TreeMap obj = new TreeMap();
5.              obj.put("A", new Integer(1));
6.              obj.put("B", new Integer(2));
7.              obj.put("C", new Integer(3));
8.              System.out.println(obj.entrySet());
9.          }
10.     }
```

a) [A, B, C] b) [1, 2, 3] c) {A=1, B=2, C=3} d) [A=1, B=2, C=3]

41. Which of these standard collection classes implements all the standard functions on list data structure?

a) Array b) LinkedList c) HashSet d) AbstractSet

42. Which of these classes implements Set interface?

a) ArrayList b) HashSet c) LinkedList d) DynamicList

43. Which of these method is used to make all elements of an equal to specified value?

a) add() b) fill() c) all() d) set().

44. Which of these method of Array class is used sort an array or its subset?

a) binarysort() b) bubblesort() c) sort() d) insert()

45. Which of these methods can be used to search an element in a list?

a) find() b) sort() c) get() d) binaryserach()

46. Which of these method is used to change an element in a LinkedList Object?

a) change() b) set() c) redo() d) add()

47. What is the output of this program?

```
1.      import java.util.*;
```

```

2.      class ArrayList {
3.          public static void main(String args[]) {
4.              ArrayList obj1 = new ArrayList();
5.              ArrayList obj2 = new ArrayList();
6.              obj1.add("A");
7.              obj1.add("B");
8.              obj2.add("A");
9.              obj2.add(1, "B");
10.             System.out.println(obj1.equals(obj2));
11.         }
12.     }

```

a) 0 b) 1 c) true d) false

48. What is the output of this program?

```

1.      import java.util.*;
2.      class Array {
3.          public static void main(String args[]) {
4.              int array[] = new int [5];
5.              for (int i = 5; i > 0; i--)
6.                  array[5 - i] = i;
7.              Arrays.sort(array);
8.              for (int i = 0; i < 5; ++i)
9.                  System.out.print(array[i]);;
10.         }
11.     }

```

a) 12345 b) 54321 c) 1234 d) 5432

49. What is the output of this program?

```

1.      import java.util.*;
2.      class Array {
3.          public static void main(String args[]) {
4.              int array[] = new int [5];

```

```
5.         for (int i = 5; i > 0; i--)
6.             array[5-i] = i;
7.         Arrays.fill(array, 1, 4, 8);
8.         for (int i = 0; i < 5 ; i++)
9.             System.out.print(array[i]);
10.        }
11.    }
```

a) 12885 b) 12845 c) 58881 d) 54881

50. What is the output of this program?

```
1.         import java.util.*;
2.         class Array {
3.             public static void main(String args[] ) {
4.                 int array[] = new int [5];
5.                 for (int i = 5; i > 0; i--)
6.                     array[5 - i] = i;
7.                 Arrays.sort(array);
8.                 System.out.print(Arrays.binarySearch(array, 4));
9.             }
10.        }
```

a) 2 b) 3 c) 4 d) 5

51. Which of these class object can be used to form a dynamic array?

a) ArrayList b) Map c) Vector d) ArrayList & Map

52. Which of these are legacy classes?

a) Stack b) Hashtable c) Vector d) All of the mentioned

53. Which of these is the interface of legacy?

a) Map b) Enumeration c) HashMap d) Hashtable

54. What is the name of data member of class Vector which is used to store number of elements in the vector?

a) length b) elements c) elementCount d) capacity

55. Which of these methods is used to add elements in vector at specific location?

a) add() b) set() c) AddElement() d) addElement()

56. What is the output of this program?

```
1.      import java.util.*;
2.      class vector {
3.          public static void main(String args[]) {
4.              Vector obj = new Vector(4,2);
5.              obj.addElement(new Integer(3));
6.              obj.addElement(new Integer(2));
7.              obj.addElement(new Integer(5));
8.              System.out.println(obj.elementAt(1));
9.          }
10.     }
```

a) 0 b) 3 c) 2 d) 5

57. What is the output of this program?

```
1.      import java.util.*;
2.      class vector {
3.          public static void main(String args[]) {
4.              Vector obj = new Vector(4,2);
5.              obj.addElement(new Integer(3));
6.              obj.addElement(new Integer(2));
7.              obj.addElement(new Integer(5));
8.              System.out.println(obj.capacity());
9.          }
10.     }
```

a) 2 b) 3 c) 4 d) 6

58. What is the output of this program?

```
1.      import java.util.*;
2.      class vector {
3.          public static void main(String args[]) {
4.              Vector obj = new Vector(4,2);
5.              obj.addElement(new Integer(3));
6.              obj.addElement(new Integer(2));
7.              obj.addElement(new Integer(5));
8.              obj.insertElementAt(new Integer(8), 2);
9.              System.out.println(obj);
10.         }
11.     }
```

a) [3, 2, 6] b) [3, 2, 8] c) [3, 2, 6, 8] d) [3, 2, 8, 6]

59. What is the output of this program?

```
1.      import java.util.*;
2.      class vector {
3.          public static void main(String args[]) {
4.              Vector obj = new Vector(4,2);
5.              obj.addElement(new Integer(3));
6.              obj.addElement(new Integer(2));
7.              obj.addElement(new Integer(5));
8.              obj.removeAll(obj);
9.              System.out.println(obj.isEmpty());
10.         }
11.     }
```

a) 0 b) 1 c) true d) false

60. What is the output of this program?

```
1.      import java.util.*;
```

```
2.      class stack {
3.          public static void main(String args[]) {
4.              Stack obj = new Stack();
5.              obj.push(new Integer(3));
6.              obj.push(new Integer(2));
7.              obj.pop();
8.              obj.push(new Integer(5));
9.              System.out.println(obj);
10.         }
11.     }
```

a) [3, 5] b) [3, 2] c) [3, 2, 5] d) [3, 5, 2]

61. Why are generics used?

- a) Generics make code more fast.
- b) Generics make code more optimised and readable.
- c) Generics add stability to your code by making more of your bugs detectable at compile time.
- d) Generics add stability to your code by making more of your bugs detectable at run time.

62. Which of these type parameters is used for a generic class to return and accept any type of object?

- a) K
- b) N
- c) T
- d) V

63. Which of these type parameters is used for a generic class to return and accept a number?

- a) K
- b) N
- c) T
- d) V

64. Which of these is an correct way of defining generic class?

- a) `class name(T1, T2, ..., Tn) { /* ... */ }`
- b) `class name { /* ... */ }`
- c) `class name[T1, T2, ..., Tn] { /* ... */ }`
- d) `class name{T1, T2, ..., Tn} { /* ... */ }`

65. Which of the following is incorrect statement regarding the use of generics and parameterized types in Java?

- a) Generics provide type safety by shifting more type checking responsibilities to the compiler.
- b) Generics and parameterized types eliminate the need for down casts when using Java Collections.
- c) When designing your own collections class (say, a linked list), generics and parameterized types allow you to achieve type safety with just a single class definition as opposed to defining multiple classes.
- d) All of the mentioned

66. Which of the following reference types cannot be generic?

- a) Anonymous inner class
- b) Interface
- c) Inner class
- d) All of the mentioned

67. What is the output of this program?

```
1.      import java.util.*;
2.      public class genericstack <E> {
3.          Stack <E> stk = new Stack <E>();
4.          public void push(E obj) {
5.              stk.push(obj);
6.          }
7.          public E pop() {
8.              E obj = stk.pop();
```

```

9.         return obj;
10.        }
11.    }
12.    class Output {
13.        public static void main(String args[]) {
14.            genericstack <String> gs = new genericstack<String>();
15.            gs.push("Hello");
16.            System.out.println(gs.pop());
17.        }
18.    }

```

- a) H
- b) Hello
- c) Runtime Error
- d) Compilation Error

68. What is the output of this program?

```

1.    import java.util.*;
2.    public class genericstack <E> {
3.        Stack <E> stk = new Stack <E>();
4.        public void push(E obj) {
5.            stk.push(obj);
6.        }
7.        public E pop() {
8.            E obj = stk.pop();
9.            return obj;
10.        }
11.    }
12.    class Output {
13.        public static void main(String args[]) {
14.            genericstack <Integer> gs = new genericstack<Integer>();
15.            gs.push(36);

```

```
16.         System.out.println(gs.pop());
17.     }
18. }
```

- a) 0
- b) 36
- c) Runtime Error
- d) Compilation Error

69. What is the output of this program?

```
1.     import java.util.*;
2.     public class genericstack <E> {
3.         Stack <E> stk = new Stack <E>();
4.         public void push(E obj) {
5.             stk.push(obj);
6.         }
7.         public E pop() {
8.             E obj = stk.pop();
9.             return obj;
10.        }
11.    }
12.    class Output {
13.        public static void main(String args[]) {
14.            genericstack <String> gs = new genericstack<String>();
15.            gs.push("Hello");
16.            System.out.print(gs.pop() + " ");
17.            genericstack <Integer> gs = new genericstack<Integer>();
18.            gs.push(36);
19.            System.out.println(gs.pop());
20.        }
21.    }
```

- a) Error
- b) Hello
- c) 36
- d) Hello 36

70. What is the output of this program?

```
1.      import java.util.*;
2.      public class genericstack <E> {
3.          Stack <E> stk = new Stack <E>();
4.          public void push(E obj) {
5.              stk.push(obj);
6.          }
7.          public E pop() {
8.              E obj = stk.pop();
9.              return obj;
10.         }
11.     }
12.     class Output {
13.         public static void main(String args[]) {
14.             genericstack <Integer> gs = new genericstack<Integer>();
15.             gs.push(36);
16.             System.out.println(gs.pop());
17.         }
18.     }
```

- a) H
- b) Hello
- c) Runtime Error
- d) Compilation Error

71. What are generic methods?

- a) Generic methods are the methods defined in a generic class.

- b) Generic methods are the methods that extend generic class's methods.
- c) Generic methods are methods that introduce their own type parameters.
- d) Generic methods are methods that take void parameters.

72. Which of these type parameters is used for a generic methods to return and accept any type of object?

- a) K
- b) N
- c) T
- d) V

73. Which of these type parameters is used for a generic methods to return and accept a number?

- a) K
- b) N
- c) T
- d) V

74. Which of these is an correct way of defining generic method?

- a) `name(T1, T2, ..., Tn) { /* ... */ }`
- b) `public name { /* ... */ }`
- c) `class name[T1, T2, ..., Tn] { /* ... */ }`
- d) `name{T1, T2, ..., Tn} { /* ... */ }`

75. Which of the following is incorrect statement regarding the use of generics and parameterized types in Java?

- a) Generics provide type safety by shifting more type checking responsibilities to the compiler.
- b) Generics and parameterized types eliminate the need for down casts when using Java Collections.
- c) When designing your own collections class (say, a linked list), generics and parameterized types allow you to achieve type safety with just a single class definition as opposed to defining multiple classes.
- d) All of the mentioned

76. Which of the following allows us to call generic methods as a normal method?

- a) Type Interface
- b) Interface

c) Inner class

d) All of the mentioned

77. What is the output of this program?

```
1.      import java.util.*;
2.      public class genericstack <E> {
3.          Stack <E> stk = new Stack <E>();
4.          public void push(E obj) {
5.              stk.push(obj);
6.          }
7.          public E pop() {
8.              E obj = stk.pop();
9.              return obj;
10.         }
11.     }
12.     class Output {
13.         public static void main(String args[]) {
14.             genericstack <String> gs = new genericstack<String>();
15.             gs.push("Hello");
16.             System.out.println(gs.pop());
17.         }
18.     }
```

a) H

b) Hello

c) Runtime Error

d) Compilation Error

78. What is the output of this program?

```
1.      import java.util.*;
2.      public class genericstack <E> {
3.          Stack <E> stk = new Stack <E>();
4.          public void push(E obj) {
5.              stk.push(obj);
```

```

6.         }
7.     public E pop() {
8.         E obj = stk.pop();
9.         return obj;
10.    }
11. }
12. class Output {
13.     public static void main(String args[]) {
14.         genericstack <Integer> gs = new genericstack<Integer>();
15.         gs.push(36);
16.         System.out.println(gs.pop());
17.     }
18. }

```

- a) 0
- b) 36
- c) Runtime Error
- d) Compilation Error

79. What is the output of this program?

```

1.     import java.util.*;
2.     public class genericstack <E> {
3.         Stack <E> stk = new Stack <E>();
4.         public void push(E obj) {
5.             stk.push(obj);
6.         }
7.         public E pop() {
8.             E obj = stk.pop();
9.             return obj;
10.        }
11.    }
12.    class Output {
13.        public static void main(String args[]) {

```

```

14.         genericstack <String> gs = new genericstack<String>();
15.         gs.push("Hello");
16.         System.out.print(gs.pop() + " ");
17.         genericstack <Integer> gs = new genericstack<Integer>();
18.         gs.push(36);
19.         System.out.println(gs.pop());
20.     }
21. }

```

- a) Error
- b) Hello
- c) 36
- d) Hello 36

80. What is the output of this program?

```

1.     import java.util.*;
2.     public class genericstack <E> {
3.         Stack <E> stk = new Stack <E>();
4.         public void push(E obj) {
5.             stk.push(obj);
6.         }
7.         public E pop() {
8.             E obj = stk.pop();
9.             return obj;
10.        }
11.    }
12.    class Output {
13.        public static void main(String args[]) {
14.            genericstack <Integer> gs = new genericstack<Integer>();
15.            gs.push(36);
16.            System.out.println(gs.pop());
17.        }

```

18.            }

- a) H
- b) Hello
- c) Runtime Error
- d) Compilation Error

81. What does AWT stands for?

- a) All Window Tools
- b) All Writing Tools
- c) Abstract Window Toolkit
- d) Abstract Writing Toolkit

82. Which of these is used to perform all input & output operations in Java?

- a) streams
- b) Variables
- c) classes
- d) Methods

83. Which of these is a type of stream in Java?

- a) Integer stream
- b) Short stream
- c) Byte stream
- d) Long stream

84. Which of these classes are used by Byte streams for input and output operation?

- a) InputStream
- b) OutputStream
- c) Reader
- d) All of the mentioned

85. Which of these classes are used by character streams for input and output operations?

- a) InputStream
- b) Writer
- c) ReadStream

d) InputStream

86. Which of these class is used to read from byte array?

a) InputStream.

b) BufferedInputStream.

c) ArrayInputStream.

d) ByteArrayInputStream.

87. What is the output of this program if input given is 'abcqfghqbcd'?

```
1.      class Input_Output {
2.          public static void main(String args[]) throws IOException {
3.              char c;
4.              BufferedReader obj = new BufferedReader(new
InputStreamReader(System.in));
5.              do {
6.                  c = (char) obj.read();
7.                  System.out.print(c);
8.              } while(c != 'q');
9.          }
10.     }
```

a) abcqfgh

b) abc

c) abcq

d) abcqfghq

88. What is the output of this program if input given is "abc'def'egh"?

```
1.      class Input_Output {
2.          public static void main(String args[]) throws IOException {
3.              char c;
4.              BufferedReader obj = new BufferedReader(new
InputStreamReader(System.in));
5.              do {
6.                  c = (char) obj.read();
```

```
7.         System.out.print(c);
8.         } while(c!="");
9.     }
10.    }
```

- a) abc'
- b) abcdef'
- c) abc'def'egh
- d) abcqfghq

89. What is the output of this program?

```
1.     class output {
2.         public static void main(String args[])
3.         {
4.             StringBuffer c = new StringBuffer("Hello");
5.             System.out.println(c.length());
6.         }
7.     }
```

- a) 4
- b) 5
- c) 6
- d) 7

90. What is the output of this program?

```
1.     class output {
2.         public static void main(String args[])
3.         {
4.             StringBuffer s1 = new StringBuffer("Hello");
5.             StringBuffer s2 = s1.reverse();
6.             System.out.println(s2);
7.         }
8.     }
```

- a) Hello
- b) olleH

c) HelloolleH

d) olleHHello

91. Which exception is thrown by read() method?

a) IOException

b) InterruptedException

c) SystemException

d) SystemInputException

92. Which of these is used to read a string from the input stream?

a) get()

b) getLine()

c) read()

d) readLine()

93. Which of these class is used to read characters and strings in Java from console?

a) BufferedReader

b) StringReader

c) BufferedStreamReader

d) InputStreamReader

94. Which of these classes are used by Byte streams for input and output operation?

a) InputStream

b) InputStream

c) Reader

d) All of the mentioned

95. Which of these class is implemented by FilterInputStream class?

a) InputStream

b) InputStream

c) BufferedInputStream

d) SequenceInputStream

96. Which of these class is used to read from a file?

- a) InputStream
- b) BufferedInputStream
- c) FileInputStream
- d) BufferedFileInputStream

97. What is the output of this program if input given is “Hello stop World”?

```
1.      class Input_Output {
2.          public static void main(String args[]) throws IOException {
3.              string str;
4.              BufferedReader obj = new BufferedReader(new
InputStreamReader(System.in));
5.              do {
6.                  str = (char) obj.readLine();
7.                  System.out.print(str);
8.              } while(!str.equals("strong"));
9.          }
10.     }
```

- a) Hello
- b) Hello stop
- c) World
- d) Hello stop World

98. What is the output of this program?

```
1.      class output {
2.          public static void main(String args[])
3.          {
4.              StringBuffer c = new StringBuffer("Hello");
5.              StringBuffer c1 = new StringBuffer(" World");
6.              c.append(c1);
7.              System.out.println(c);
8.          }
```

9.            }

- a) Hello
- b) World
- c) Helloworld
- d) Hello World

99. What is the output of this program?

```
1.            class output {  
2.                public static void main(String args[])  
3.                {  
4.                    StringBuffer s1 = new StringBuffer("Hello");  
5.                    s1.setCharAt(1,x);  
6.                    System.out.println(s1);  
7.                }  
8.            }
```

- a) xello
- b) xxxxx
- c) Hxlllo
- d) Hexlo

100. What is the output of this program if input given is "abc'def'egh"?

```
1.            class Input_Output {  
2.                public static void main(String args[]) throws IOException {  
3.                    char c;  
4.                    BufferedReader obj = new BufferedReader(new  
                  InputStreamReader(System.in));  
5.                    do {  
6.                        c = (char) obj.read();  
7.                        System.out.print(c);  
8.                    } while(c != '\n');  
9.                }  
10.            }
```

- a) abc'
- b) abcdef'
- c) abc'def' egh
- d) abcqfghq

101. Which of these class contains the methods print() & println()?

- a) System
- b) System.out
- c) BUfferedOutputStream
- d) PrintStream

102. Which of these methods can be used to writing console output?

- a) print()
- b) println()
- c) write()
- d) All of the mentioned

103. Which of these class is used to create an object whose character sequence is mutable?

- a) String()
- b) StringBuffer()
- c) Both of the mentioned
- d) None of the mentioned

104. Which of these method of class StringBuffer is used to reverse sequence of characters?

- a) reverse()
- b) reverseall()
- c) Reverse()
- d) reverseAll()

105. Which of these classes are used by character streams output operations?

- a) InputStream

- b) Writer
- c) ReadStream
- d) InputStream

106. Which of the following statement is correct?

- a) reverse() method reverses all characters.
- b) reverseall() method reverses all characters.
- c) replace() method replaces first occurrence of a character in invoking string with another character.
- d) replace() method replaces last occurrence of a character in invoking string with another character.

107. What is the output of this program?

```
1.      class output {
2.          public static void main(String args[])
3.          {
4.              String a="hello i love java";
5.              System.out.println(indexof('i')+" "+indexOf('o')+" "+lastIndexOf('i')+"
"+lastIndexOf('o') ));
6.          }
7.      }
```

- a) 6 4 6 9
- b) 5 4 5 9
- c) 7 8 8 9
- d) 4 3 6 9

108. What is the output of this program?

```
1.      class output {
2.          public static void main(String args[])
3.          {
4.              StringBuffer s1 = new StringBuffer("Hello");
```

```
5.         StringBuffer s2 = s1.reverse();
6.         System.out.println(s2);
7.     }
8. }
```

- a) Hello
- b) olleH
- c) HelloolleH
- d) olleHHello

109. What is the output of this program?

```
1.     class output {
2.         public static void main(String args[])
3.     {
4.         StringBuffer s1 = new StringBuffer("Hello World");
5.         s1.insert(6 , "Good ");
6.         System.out.println(s1);
7.     }
8. }
```

- a) HelloGoodWorld
- b) HellGoodoWorld
- c) HellGood oWorld
- d) Hello Good World

110. What is the output of this program?

```
1.     class output {
2.         public static void main(String args[])
3.     {
4.         char c[]={ 'a','1','b',' ','A','0' };
5.         for (int i = 0; i < 5; ++i) {
6.             if(Character.isDigit(c[i]))
7.                 System.out.println(c[i]" is a digit");
```

```
8.         if(Character.isWhitespace(c[i]))
9.             System.out.println(c[i] " is a Whitespace character");
10.        if(Character.isUpperCase(c[i]))
11.            System.out.println(c[i] " is an Upper case Letter");
12.        if(Character.isUpperCase(c[i]))
13.            System.out.println(c[i] " is a lower case Letter");
14.            i = i + 3;
15.        }
16.    }
17. }
```

- a) a is a lower case Letter  
is White space character
- b) b is a lower case Letter  
is White space characte
- c) a is a lower case Letter  
A is a upper case Letter
- d) a is a lower case Letter  
0 is a digit

111. Which of these class contains the methods used to write in a file?

- a) FileStream
- b) FileInputStream
- c) BUfferedOutputStream
- d) FileBufferStream

112. Which of these exception is thrown in cases when the file specified for writing it not found?

- a) IOException
- b) FileException
- c) FileNotFoundException
- d) FileInputException

113. Which of these methods are used to read in from file?

- a) get()
- b) read()
- c) scan()
- d) readFileInput()

114. Which of these values is returned by read() method is end of file (EOF) is encountered?

- a) 0
- b) 1
- c) -1
- d) Null

115. Which of these exception is thrown by close() and read() methods?

- a) IOException
- b) FileNotFoundException
- c) FileInputOutputException
- d) FileException

116. Which of these methods is used to write() into a file?

- a) put()
- b) putFile()
- c) write()
- d) writeFile()

117. What is the output of this program?

```
1.      import java.io.*;
2.      class filesinputoutput {
3.          public static void main(String args[]) {
4.              InputStream obj = new FileInputStream("inputoutput.java");
5.              System.out.print(obj.available());
6.          }
```

7.                    }

Note: inputoutput.java is stored in the disk.

- a) true
- b) false
- c) prints number of bytes in file
- d) prints number of characters in the file

118. What is the output of this program?

```
1.            import java.io.*;
2.            public class filesinputoutput {
3.                public static void main(String[] args) {
4.                    String obj = "abc";
5.                    byte b[] = obj.getBytes();
6.                    ByteArrayInputStream obj1 = new ByteArrayInputStream(b);
7.                    for (int i = 0; i < 2; ++ i) {
8.                        int c;
9.                        while((c = obj1.read()) != -1) {
10.                            if(i == 0) {
11.                                System.out.print(Character.toUpperCase((char)c));
12.                                obj2.write(1);
13.                                }
14.                                }
15.                            System.out.print(obj2);
16.                            }
17.                        }
18.                    }
```

- a) AaBaCa
- b) ABCaaa
- c) AaaBaaCaa
- d) AaBaaCaaa

119. What is the output of this program?

```
1.      import java.io.*;
2.      class Chararrayinput {
3.          public static void main(String[] args) {
4.              String obj = "abcdef";
5.              int length = obj.length();
6.              char c[] = new char[length];
7.              obj.getChars(0, length, c, 0);
8.              CharArrayReader input1 = new CharArrayReader(c);
9.              CharArrayReader input2 = new CharArrayReader(c, 0, 3);
10.             int i;
11.             try {
12.                 while((i = input2.read()) != -1) {
13.                     System.out.print((char)i);
14.                 }
15.             }
16.             catch (IOException e) {
17.                 e.printStackTrace();
18.             }
19.         }
20.     }
```

- a) abc
- b) abcd
- c) abcde
- d) abcdef

120. What is the output of this program?

```
1.      import java.io.*;
2.      class Chararrayinput {
3.          public static void main(String[] args) {
4.              String obj = "abcdefgh";
```

```

5.         int length = obj.length();
6.         char c[] = new char[length];
7.         obj.getChars(0, length, c, 0);
8.         CharArrayReader input1 = new CharArrayReader(c);
9.         CharArrayReader input2 = new CharArrayReader(c, 1, 4);
10.        int i;
11.        int j;
12.        try {
13.            while((i = input1.read()) == (j = input2.read())) {
14.                System.out.print((char)i);
15.            }
16.        }
17.        catch (IOException e) {
18.            e.printStackTrace();
19.        }
20.    }
21.    }

```

- a) abc
- b) abcd
- c) abcde
- d) None of the mentioned

.

121. Which of these packages contain classes and interfaces used for input & output operations of a program?

- a) java.util
- b) java.lang
- c) java.io
- d) All of the mentioned

121. Which of these class is not a member class of java.io package?

- a) String

- b) StringReader
- c) Writer
- d) File

.

122. Which of these interface is not a member of java.io package?

- a) DataInput
- b) ObjectInput
- c) ObjectFilter
- d) FileFilter

123. Which of these class is not related to input and output stream in terms of functioning?

- a) File
- b) Writer
- c) InputStream
- d) Reader

124. Which of these is specified by a File object?

- a) a file in disk
- b) directory path
- c) directory in disk
- d) None of the mentioned

125. Which of these is method for testing whether the specified element is a file or a directory?

- a) IsFile()
- b) isFile()
- c) Isfile()
- d) isfile()

126. What is the output of this program?

- 1. **import** java.io.\*;
- 2. **class** files {

```
3.      public static void main(String args[]) {
4.          File obj = new File("/java/system");
5.          System.out.print(obj.getName());
6.      }
7.  }
```

- a) java
- b) system
- c) java/system
- d) /java/system

127. What is the output of this program?

```
1.      import java.io.*;
2.      class files {
3.          public static void main(String args[]) {
4.              File obj = new File("/java/system");
5.              System.out.print(obj.getAbsolutePath());
6.          }
7.      }
```

Note: file is made in c drive.

- a) java
- b) system
- c) java/system
- d) /java/system

128. What is the output of this program?

```
1.      import java.io.*;
2.      class files {
3.          public static void main(String args[]) {
4.              File obj = new File("/java/system");
5.              System.out.print(obj.canWrite());
6.              System.out.print(" " + obj.canRead());
```

```
7.         }
8.         }
```

Note: file is made in c drive.

- a) true false
- b) false true
- c) true true
- d) false false

129. What is the output of this program?

```
1.         import java.io.*;
2.         class files {
3.             public static void main(String args[]) {
4.                 File obj = new File("/java/system");
5.                 System.out.print(obj.getParent());
6.                 System.out.print(" " + obj.isFile());
7.             }
8.         }
```

Note: file is made in c drive.

- a) java true
- b) java false
- c) \java false
- d) \java true

130. Which of these classes is used for input and output operation when working with bytes?

- a) InputStream
- b) Reader
- c) Writer
- d) All of the mentioned

131. Which of these class is used to read and write bytes in a file?

- a) FileReader

- b) FileWriter
- c) FileInputStream
- d) InputStreamReader

132 Which of these method of InputStream is used to read integer representation of next available byte input?

- a) read()
- b) scanf()
- c) get()
- d) getInteger()

133. Which of these data type is returned by every method of OutputStream?

- a) int
- b) float
- c) byte
- d) None of the mentioned

134. Which of these is a method to clear all the data present in output buffers?

- a) clear()
- b) flush()
- c) fflush()
- d) close()

.

135. Which of these is method is used for writing bytes to an outputstream?

- a) put()
- b) print()
- c) printf()
- d) write()

136. What is the output of this program?

1. **import** java.io.\*;

```
2.      class filesinputoutput {
3.          public static void main(String args[]) {
4.              InputStream obj = new FileInputStream("inputoutput.java");
5.              System.out.print(obj.available());
6.          }
7.      }
```

Note: inputoutput.java is stored in the disk.

- a) true
- b) false
- c) prints number of bytes in file
- d) prints number of characters in the file

137. What is the output of this program?

```
1.      import java.io.*;
2.      public class filesinputoutput {
3.          public static void main(String[] args) {
4.              String obj = "abc";
5.              byte b[] = obj.getBytes();
6.              ByteArrayInputStream obj1 = new ByteArrayInputStream(b);
7.              for (int i = 0; i < 2; ++ i) {
8.                  int c;
9.                  while ((c = obj1.read()) != -1) {
10.                     if(i == 0) {
11.                         System.out.print((char)c);
12.                     }
13.                 }
14.             }
15.         }
16.     }
```

- a) abc
- b) ABC

- c) ab
- d) AB

138. What is the output of this program?

```
1.      import java.io.*;
2.      public class filesinputoutput {
3.          public static void main(String[] args) {
4.              String obj = "abc";
5.              byte b[] = obj.getBytes();
6.              ByteArrayInputStream obj1 = new ByteArrayInputStream(b);
7.              for (int i = 0; i < 2; ++ i) {
8.                  int c;
9.                  while ((c = obj1.read()) != -1) {
10.                     if (i == 0) {
11.                         System.out.print(Character.toUpperCase((char)c));
12.                     }
13.                 }
14.             }
15.         }
16.     }
```

- a) abc
- b) ABC
- c) ab
- d) AB

139. What is the output of this program?

```
1.      import java.io.*;
2.      public class filesinputoutput {
3.          public static void main(String[] args) {
4.              String obj = "abc";
5.              byte b[] = obj.getBytes();
```

```
6.      ByteArrayInputStream obj1 = new ByteArrayInputStream(b);
7.      for (int i = 0; i < 2; ++ i) {
8.          int c;
9.          while ((c = obj1.read()) != -1) {
10.             if (i == 0) {
11.                 System.out.print(Character.toUpperCase((char)c));
12.                 obj2.write(1);
13.             }
14.         }
15.         System.out.print(obj2);
16.     }
17. }
18. }
```

- a) AaBaCa
- b) ABCaaa
- c) AaaBaaCaa
- d) AaBaaCaaa

140. Which of these stream contains the classes which can work on character stream?

- a) InputStream
- b) OutputStream
- c) Character Stream
- d) All of the mentioned

141. Which of these class is used to read characters in a file?

- a) FileReader
- b) FileWriter
- c) FileInputStream
- d) InputStreamReader

142. Which of these method of FileReader class is used to read characters from a file?

- a) read()
- b) scanf()
- c) get()
- d) getInteger()

143. Which of these class can be used to implement input stream that uses a character array as the source?

- a) BufferedReader
- b) FileReader
- c) CharArrayReader
- d) FileArrayReader

144. Which of these is a method to clear all the data present in output buffers?

- a) clear()
- b) flush()
- c) fflush()
- d) close()

145. Which of these classes can return more than one character to be returned to input stream?

- a) BufferedReader
- b) Bufferedwriter
- c) PushbachReader
- d) CharArrayReader

146. What is the output of this program?

```
1.      import java.io.*;
2.      class filesinputoutput {
3.          public static void main(String args[]) {
4.              InputStream obj = new FileInputStream("inputoutput.java");
5.              System.out.print(obj.available());
```

```
6.         }
7.         }
```

Note: inputoutput.java is stored in the disk.

- a) true
- b) false
- c) prints number of bytes in file
- d) prints number of characters in the file

147. What is the output of this program?

```
1.         import java.io.*;
2.         class Chararrayinput {
3.             public static void main(String[] args) {
4.                 String obj = "abcdef";
5.                 int length = obj.length();
6.                 char c[] = new char[length];
7.                 obj.getChars(0,length,c,0);
8.                 CharArrayReader input1 = new CharArrayReader(c);
9.                 CharArrayReader input2 = new CharArrayReader(c, 0, 3);
10.                int i;
11.                try {
12.                    while ((i = input1.read()) != -1) {
13.                        System.out.print(((char)i));
14.                    }
15.                }
16.                catch (IOException e) {
17.                    // TODO Auto-generated catch block
18.                    e.printStackTrace();
19.                }
20.            }
21.        }
```

- a) abc
- b) abcd
- c) abcde
- d) abcdef

148. What is the output of this program?

```
1.      import java.io.*;
2.      class Chararrayinput {
3.          public static void main(String[] args) {
4.              String obj = "abcdef";
5.              int length = obj.length();
6.              char c[] = new char[length];
7.              obj.getChars(0, length, c, 0);
8.              CharArrayReader input1 = new CharArrayReader(c);
9.              CharArrayReader input2 = new CharArrayReader(c, 0, 3);
10.         int i;
11.         try {
12.             while ((i = input2.read()) != -1) {
13.                 System.out.print((char)i);
14.             }
15.         }
16.         catch (IOException e) {
17.             // TODO Auto-generated catch block
18.             e.printStackTrace();
19.         }
20.     }
21. }
```

- a) abc
- b) abcd
- c) abcde

d) abcdef

149. What is the output of this program?

```
1.      import java.io.*;
2.      class Chararrayinput {
3.          public static void main(String[] args) {
4.              String obj = "abcdefgh";
5.              int length = obj.length();
6.              char c[] = new char[length];
7.              obj.getChars(0, length, c, 0);
8.              CharArrayReader input1 = new CharArrayReader(c);
9.              CharArrayReader input2 = new CharArrayReader(c, 1, 4);
10.             int i;
11.             int j;
12.             try {
13.                 while ((i = input1.read()) == (j = input2.read())) {
14.                     System.out.print((char)i);
15.                 }
16.             }
17.             catch (IOException e) {
18.                 // TODO Auto-generated catch block
19.                 e.printStackTrace();
20.             }
21.         }
22.     }
```

a) abc

b) abcd

c) abcde

d) None of the mentioned

**UNIT-5**

1. Which of the following classes are not in the java.awt package? [ ]
  - a. Color b. Font c. Component d. JFrame
2. The interface \_\_\_\_\_ should be implemented to listen for a button action event.[ ]
  - a. MouseListener b. ActionListener c. FocusListener d. WindowListener
- 3 Classes for reading and writing(input and output) are in \_\_\_\_\_ package. [ ]
  - a. java.lang b. java.io c. java.net d. java.net
4. \_\_\_\_\_ package contains classes and interfaces that are used in enabling programs to encrypt data and control the access privileges.
  - a. java.lang b. java.io c. java.net d. java.security
- 5.A \_\_\_\_\_ method cannot be overridden by subclasses.
6. \_\_\_\_\_ is the capability of a class to use the properties and methods of another class while adding its own functionality.
7. An \_\_\_\_\_ is a named collection of method definitions without implementations. 8. \_\_\_\_\_ is an ordered list of directories or ZIP files in which to search for class files.
9. \_\_\_\_\_ is a collection of related classes and interfaces providing access protection and namespace
- 13.
10. Fundamental classes are defined in \_\_\_\_\_ package.
11. The \_\_\_\_\_ statement occurs immediately after the package statement and before the class statement:
  - a. import b. export c.a&b d. none
12. \_\_\_\_\_ of a class is enabled by the class implementing the java.io.Serializable interface.
13. \_\_\_\_\_ interface provides for reading bytes from a binary stream and reconstructing from the data in any of the Java primitive types.
14. \_\_\_\_\_ instance is written in the serialization stream and it is the responsibility of the class to save and restore the contents of its instances.

15. An \_\_\_\_\_ is an object that describes a state change in a source.  
 A) event                      B) circular                      C) object                      D) Class
16. Frames, Panels and objects that can contain other GUI objects are known as \_\_\_\_\_ [ ]  
 a. Container b. Tab c. Dialog d. Layout
17. \_\_\_\_\_ are referred to as heavyweight components
18. \_\_\_\_\_ is a component that presents a hierarchical view of the data.
19. \_\_\_\_\_ manages to store several different layouts
20. . The interface \_\_\_\_\_ should be implemented to listen for a button action event. [ ]  
 a. MouseListener b. ActionListener c. FocusListener d. WindowListener
21. To determine the width and height of an application which of the following method is used? [ ]  
 a) getWidthHeight()                      b) setWidthHeight()                      c) getSize()                      d) setHeightWidth()
22. What are the applet's lifecycle methods?  
 a) init(), start() and destroy()  
 c) init(), start(), paint(), stop() and destroy()
23. The preferredSize property is ignored in \_\_\_\_\_ [ ]  
 a) FlowLayout                      b) GridLayout                      c) BorderLayout                      d) For any layout
24. \_\_\_\_\_ is the process of confining paint operations to a limited area or shape.
25. The \_\_\_\_\_ interface should be implemented to listen for a button action event.
26. \_\_\_\_\_ cannot be shared by a GUI component.

27. Which of the following objects are input to the paint() method?

- a) canvas                      b) graphics
- c) paint                        d) image

28. \_\_\_\_\_ method is called on calling the repaint method.

29. \_\_\_\_\_ contains and controls the layout

- B) cookie
- c. component
- A) object                      d. container                      C) Container

30. Servlet is a java \_\_\_\_\_ side program.

- A) object    b. sever    c. Class    d. none

31. A JMenu is a not subclass of \_\_\_\_\_ . [ ]

- a. JMenuItem    b. AbstractButton    c. JComponent    d. JButton

32.. Swing components that don't rely on native GUI are referred to as \_\_\_\_\_ . [ ]

- a. lightweight components    b. heavyweight components
- c. GUI components    d. Non-GUI components

33. \_\_\_\_\_ method will be called when an applet begins its execution

34. The combination of a text field and a drop down list is \_\_\_\_\_

35. To add a component c to a JPanel p, use \_\_\_\_\_

36. Which of the following properties are in JApplet? [ ]

- a. contentPane    b. iconImage    c. resizable    d. title

37. The default layout out of a JPanel is \_\_\_\_\_ [ ]

- a) FlowLayout                      b) GridLayout                      c) BorderLayout                      d) default Layout

38. You should override the \_\_\_\_\_ method to draw things on a Swing component.

- a) repaint()                      b) update()                      c) paintComponent()                      d) init()

39. Which of these packages contains all the classes and methods required for event handling in Java?

- a) java.applet
- b) java.awt
- c) java.event
- d) java.awt.event

40. What is an event in delegation event model used by Java programming language?

- a) An event is an object that describes a state change in a source.
- b) An event is an object that describes a state change in processing.
- c) An event is an object that describes any change by the user and system.
- d) An event is a class used for defining object, to create events.

41. Which of these methods are used to register a keyboard event listener?

- a) KeyListener()
- b) addKistener()
- c) addKeyListener()
- d) eventKeyboardListener()

42. Which of these methods are used to register a mouse motion listener?

- a) addMouse()
- b) addMouseListener()
- c) addMouseMotionListner()
- d) eventMouseMotionListener()

43. What is a listener in context to event handling?

- a) A listener is a variable that is notified when an event occurs.
- b) A listener is a object that is notified when an event occurs.
- c) A listener is a method that is notified when an event occurs.
- d) None of the mentioned

44. Event class is defined in which of these libraries?

- a) java.io
- b) java.lang
- c) java.net
- d) java.util

45. Which of these methods can be used to determine the type of event?

- a) getID()
- b) getSource()
- c) getEvent()
- d) getEventObject()

46. Which of these class is super class of all the events?

- a) EventObject
- b) EventClass
- c) ActionEvent
- d) ItemEvent

47. Which of these events will be notified if scroll bar is manipulated?

- a) ActionEvent
- b) ComponentEvent
- c) AdjustmentEvent
- d) WindowEvent

48. Which of these events will be generated if we close an applet's window?

- a) ActionEvent
- b) ComponentEvent
- c) AdjustmentEvent
- d) WindowEvent

49. Which of these packages contains all the event handling interfaces?

- a) java.lang
- b) java.awt
- c) java.awt.event
- d) java.event

50. Which of these interfaces handles the event when a component is added to a container?

- a) ComponentListener
- b) ContainerListener
- c) FocusListener
- d) InputListener

51. Which of these interfaces define a method actionPerformed()?

- a) ComponentListener
- b) ContainerListener
- c) ActionListener
- d) InputListener

52. Which of these interfaces define four methods?

- a) ComponentListener
- b) ContainerListener
- c) ActionListener
- d) InputListener

53. Which of these interfaces define a method itemStateChanged()?

- a) ComponentListener
- b) ContainerListener
- c) ActionListener
- d) ItemListener

54. Which of these methods will respond when you click any button by mouse?

- a) mouseClicked()
- b) mouseEntered()
- c) mousePressed()
- d) All of the mentioned

55. Which of these methods will be invoked if a character is entered?

- a) keyPressed()
- b) keyReleased()
- c) keyTyped()
- d) keyEntered()

56. Which of these methods is defined in MouseMotionAdapter class?

- a) mouseDragged()
- b) mousePressed()
- c) mouseReleased()
- d) mouseClicked()

57. Which of these are constants defined in WindowEvent class?

- a) WINDOW\_ACTIVATED
- b) WINDOW\_CLOSED
- c) WINDOW\_DEICONIFIED
- d) All of the mentioned

58. Which of these is superclass of all Adapter classes?

- a) Applet
- b) ComponentEvent
- c) Event
- d) InputEvent

59. Which of these functions is called to display the output of an applet?

- a) display()
- b) print()
- c) displayApplet()
- d) PrintApplet()

60. Which of these methods can be used to output a string in an applet?

- a) display()
- b) print()
- c) drawString()
- d) transient()

61. What does AWT stand for?

- a) All Window Tools
- b) All Writing Tools
- c) Abstract Window Toolkit
- d) Abstract Writing Toolkit

62. Which of these methods is a part of Abstract Window Toolkit (AWT) ?

- a) display()
- b) print()
- c) drawString()
- d) transient()

63. Which of these modifiers can be used for a variable so that it can be accessed from any thread or parts of a program?

- a) transient
- b) volatile
- c) global
- d) No modifier is needed

64. Which of these operators can be used to get run time information about an object?

- a) getInfo
- b) Info
- c) instanceof
- d) getinfoof

65. What is the Message is displayed in the applet made by this program?

```
1.      import java.awt.*;
2.      import java.applet.*;
3.      public class myapplet extends Applet {
4.          public void paint(Graphics g) {
5.              g.drawString("A Simple Applet", 20, 20);
6.          }
7.      }
```

- a) A Simple Applet
- b) A Simple Applet 20 20
- c) Compilation Error
- d) Runtime Error

66. What is the length of the application box made by this program?

```
1.      import java.awt.*;
2.      import java.applet.*;
3.      public class myapplet extends Applet {
4.          public void paint(Graphics g) {
5.              g.drawString("A Simple Applet", 20, 20);
6.          }
7.      }
```

- a) 20
- b) 50
- c) 100

d) System dependent

67. What is the length of the application box made by this program?

```
1.      import java.awt.*;
2.      import java.applet.*;
3.      public class myapplet extends Applet {
4.          Graphic g;
5.          g.drawString("A Simple Applet", 20, 20);
6.      }
```

- a) 20
- b) Default value
- c) Compilation Error
- d) Runtime Error

68. What is the output of this program?

```
1.      import java.io.*;
2.      class Chararrayinput {
3.          public static void main(String[] args) {
4.              String obj = "abcdefgh";
5.              int length = obj.length();
6.              char c[] = new char[length];
7.              obj.getChars(0, length, c, 0);
8.              CharArrayReader input1 = new CharArrayReader(c);
9.              CharArrayReader input2 = new CharArrayReader(c, 1, 4);
10.             int i;
11.             int j;
12.             try {
13.                 while((i = input1.read()) == (j = input2.read())) {
14.                     System.out.print((char)i);
15.                 }
16.             }
```

```
17.         catch (IOException e) {  
18.             e.printStackTrace();  
19.         }  
20.     }  
21. }
```

- a) abc
- b) abcd
- c) abcde
- d) None of the mentioned

## 20 Tutorial problems

### Tutorial-1

1. Write a Java Program to print prime numbers up to a given number.
2. Write a Java Program To print Fibonacci sequence up to a given number.
3. Write a Java Program to Overload the method deposit ( ) in bank account class containing Account number account name and balance .

### Tutorial-2

1. Write a Java Program to demonstrate Widening conversion.
2. Write a Java Program to demonstrate Method chaining.
3. Write a Java Program to demonstrate String tokenizer class methods.

### Tutorial-3

1. Write a Java Program to demonstrate Dynamic method dispatch (class program).

2. Write a Java Program to check whether a String is palindrome or not.
3. Write a Java Program to Count number of words in a given text.
4. Write a Java Program To print Fibonacci sequence up to a given number.
5. Write a Java Program to demonstrate Final keyword.
6. Design and develop how to create a package and how to import the packages in java.

#### **Tutorial-4**

1. Write a Java Program to demonstrate Thread priority.
2. Write a Java Program to simulate a simple calculator
3. Write a Java Program to demonstrate Thread priority.

#### **Tutorial-5**

1. Design and develop a java program to how to handle ArithmeticException and ArrayIndexOutOfBoundsException by use try, catch and finally blocks.
2. Design and develop a java program to create three threads the first thread prints 'good morning' for every one second ,the second thread prints 'hello!' for every 2 seconds and the third thread prints 'welcome' for every three seconds
3. Design and develop a java program to show the use of throw keyword for throwing NullPointerException.

#### **Tutorial-6**

1. Design and develop a java program that reads a filename and displays the file on screen.
2. Design and develop a java program that reads a filename from keyboard and display the number of characters in the file.
3. Write a java program to illustrate Stack class.

### **Tutorial-7**

1. Write a Java Program to demonstrate List.
2. Write a Java Program to demonstrate Checkbox.
3. Write a java program to demonstrate passing parameters to Applets.
4. Develop an applet that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the button named “Compute” is clicked.

### **Tutorial-8**

1. What is a BorderLayout manager? Write a java program to try out border Layout manager.
2. What is Mouse listener interface? Explain any four Mouse listener interface methods.
3. What is a JRadioButton and Write any three constructors.
4. Create a JApplet that displays two JRadiobuttons and a JTextField and whenever a radiobutton is clicked, its text is displayed in TextField.

### **Tutorial -9**

1. Design and develop a java program to demonstrate the creation of thread groups.
2. Design and develop a java program to create multiple threads and make the threads to act on a single object using Synchronization block .
3. Design and develop a java program to show how to handle checked exceptions.
4. Design and develop a java program to demonstrate on inter-thread communication for producer-consumer problem.
5. Write a Java program for handling mouse events.

## 21 Known gaps ,if any and inclusion of the same in lecture schedule

- 1) OOP using C++

## 22 Discussion topics , if any

- 1) Difference between procedure and object oriented
- 2) C++ , c, java
- 3) Applets, servlets
- 4) C files and Java streams
- 5) JDBC Drivers

## 23 References, Journals, websites and E-links if any

- 1) Type java or topic in google search engine you will get different websites.

<http://docs.oracle.com/javase/tutorial/java/>

<http://www.freejavaguide.com/corejava.htm>

<http://iiti.ac.in/people/~tanimad/JavaTheCompleteReference.pdf>

<http://www.oracle.com/technetwork/java/newtojava/java8book-2172125.pdf>

<http://javabeginnerstutorial.com/core-java/>

<http://beginnersbook.com/java-tutorial-for-beginners-with-examples/>

## 24 Quality Measurement Sheets

- a. Course End Survey
- b. Teaching Evaluation

## 25 Student List

GEETHANJALI COLLEGE OF ENGINEERING & ENGINEERING		
Cheeryal (V), Keesara (M). R.R.Dist, A.P 501 301		
Report - Student Details		
Batch: 2014		

SINo	AdmnNo	StudentName
<b>Class / Section: CSE 22A</b>		
1	13R11A05G3	SRIKANTH AKKANAPALLY
2	14R11A0501	AKHILESH G
3	14R11A0502	AKULA KRISHNA
4	14R11A0503	ALLAM USHA SREE
5	14R11A0504	ANANT SRIVATS
6	14R11A0505	AYYAGARI V S V VARDHAMAN
7	14R11A0506	B MOHANA SREYA
8	14R11A0507	B PRASHANTH
9	14R11A0508	BANDA GAYATHRI VEENA
10	14R11A0509	BATTAR SRIKANTH RAGHAVA
11	14R11A0510	BHONSLE SAI KRUTHI
12	14R11A0511	BONAGIRI TEJASHWINI
13	14R11A0512	BOTLA MOUNIKA
14	14R11A0513	D VENKATESH
15	14R11A0514	DASARI PRANAY KUMAR
16	14R11A0515	DEEPIKA TUDIMILA
17	14R11A0516	DOKKA DIVYA
18	14R11A0517	E MADHU BHARGAVA
19	14R11A0518	K JWALA
20	14R11A0519	K SAMHITHA REDDY
21	14R11A0520	K THAPASVI REDDY
22	14R11A0521	KALAKONDA VAISHNAVI
23	14R11A0522	KAMPALLY YAMINI
24	14R11A0523	KONTHAM RAVITEJA REDDY
25	14R11A0524	KOSARAJU LEELA KRISHNA
26	14R11A0525	KRISHNA SWETHA R
27	14R11A0526	KUNDUR ROHAN REDDY
28	14R11A0527	KUNNUMAL MAVILA AMRITHA
29	14R11A0528	KURMA BHARGAV
30	14R11A0529	KUTHURU PRIYANKA
31	14R11A0530	KYASNOORI SHIVANI
32	14R11A0531	M K SREE HARSHA
33	14R11A0532	M POOJA
34	14R11A0533	M SAI KRISHNA
35	14R11A0534	M SHANMUGHA PRIYA
36	14R11A0535	MALLADI RAMYA
37	14R11A0536	MEESALA SHAILAJA
38	14R11A0537	N SRI MANASVI
39	14R11A0538	NADENDLA VENKATA VINEET MURARI
40	14R11A0539	NAKKA AKHIL
41	14R11A0540	NALLANAGULA BHANU PRAKASH

42	14R11A0541	P SOWMYA
43	14R11A0542	PARVATHALA CHETANA
44	14R11A0543	PENDYALA HEMANTH SAI
45	14R11A0544	R CHANDRA MOHAN
46	14R11A0545	REKULA CHANDRAHASA
47	14R11A0546	RUDRARAJU PAVANI
48	14R11A0547	S SWATHI
49	14R11A0548	SAI SINDHU BEERAM
50	14R11A0549	SOPPADANDI AISHWARYA RAJ
51	14R11A0550	SYED VAQUAS ASHRAF
52	14R11A0551	TALLURI MURALI KRISHNA
53	14R11A0552	TAMMEWAR SNEHIT
54	14R11A0553	THIRIVEEDI DHEERAJ KUMAR
55	14R11A0554	THUMMA VINEETHA REDDY
56	14R11A0555	VALLURU VENUMADHURI
57	14R11A0556	VAMANA GUNTLA MANOJ
58	14R11A0557	VENKATA SAITEJA Y
59	14R11A0558	VUSAKA VIKAS REDDY
60	14R11A0559	Y JYOTSNA

Total: 60 Males: 29 Females: 31

**Class / Section: CSE 22B**

1	13R11A0561	BATTU SHIRISH KUMAR REDDY
2	14R11A0560	AKELLA SUBRAMANYA SOUJANYA
3	14R11A0561	ALEENA CHACKO K
4	14R11A0562	BARLA PRAVALIKA
5	14R11A0563	BATHINI DIVYA
6	14R11A0564	BOLLA NAVEEN REDDY
7	14R11A0565	CHAVALI SWATHI
8	14R11A0566	CHEPYALA AKHIL
9	14R11A0567	CHITTARVU LAKSHMI ISHWARYA
10	14R11A0568	DAMAGALLA KARTIK
11	14R11A0569	DHATRI NAIDU BHOGADI
12	14R11A0571	DODDAPANENI SANOJ
13	14R11A0572	DUGGISHETTY SAI PRASANNA
14	14R11A0573	E PAVAN CHANDRA
15	14R11A0574	G BHAVANA
16	14R11A0575	G KISHAN
17	14R11A0576	G LASYA PRIYA
18	14R11A0577	G SIRIVENNELA
19	14R11A0578	GANAPATHI RAJU MADHURI
20	14R11A0579	GANGAVARAPU MANASA

21	14R11A0580	GATTOJI HARITHA
22	14R11A0581	GAYATHRI MANDAL
23	14R11A0582	GORTI SANTOSH KUMAR
24	14R11A0583	HRITIK S
25	14R11A0584	KALYANAM KRUTHIKA REDDY
26	14R11A0585	KANUMURI SRI PRATHYUSHA
27	14R11A0586	KOLLU VINESH BABU
28	14R11A0587	KONDISETTI NIRANJANA KUMARI
29	14R11A0588	KONIKA VENKATA PADMA PRIYA HASINI
30	14R11A0589	KURUP MEGHANA
31	14R11A0590	MADARAJU RAMYA SAI
32	14R11A0591	MEDIPALLY SRIYA
33	14R11A0592	MUDDU DEEPTHI
34	14R11A0593	N LAHARI
35	14R11A0594	N VIKHYAT
36	14R11A0595	NAKEERTHA SANDHYA RANI
37	14R11A0596	NALLAMILLI JYOTHI
38	14R11A0597	NALLAPANENI ADITYA SAI
39	14R11A0598	NAMBURI MANISHA
40	14R11A0599	NIKHIL SINGH P
41	14R11A05A0	ORUGANTI KALPANA
42	14R11A05A1	PATLORI SAI KUMAR
43	14R11A05A2	PEESAPATI VENKATA SAI VAMSI
44	14R11A05A3	PINDIPOLU SRAVYA
45	14R11A05A4	PRANATHI KRISHNA SANGANI
46	14R11A05A5	PUDHOTA AVINASH
47	14R11A05A6	S ASHA LAKSHMI
48	14R11A05A7	S CHANDANA REDDY
49	14R11A05A8	SHAIK HAFEEZ HUSSAIN
50	14R11A05A9	SHEELAM SUSHMA
51	14R11A05B0	SRAVIKA SANKU
52	14R11A05B1	THUMMALA RISHIKA REDDY
53	14R11A05B2	VARSHA CHAHAL
54	14R11A05B3	VEERAVALLI SHILPA
55	14R11A05B4	VUPPUTTURI SUSHANTH KUMAR
56	14R11A05B5	Y ADITHYA
57	14R11A05B6	YEDDU SHASHI KUMAR
58	14R11A05B7	KATAMONI HARSHITHA
59	15R15A0501	B VINESH
60	15R15A0502	B SHRAVAN KUMAR
Total: 60 Males: 22 Females: 38		

Class / Section: CSE 22C		
1	14R11A05C0	A S SRUJAN SAI
2	14R11A05C1	AITHA RANJEETH KUMAR
3	14R11A05C2	ALIGETI MAHITHA
4	14R11A05C3	BAGGU VISHNU SAI PRASAD
5	14R11A05C4	BARGELA PRANAY RAJ
6	14R11A05C5	BIROJU SAI SUGANDH CHARY
7	14R11A05C6	G AKHILA
8	14R11A05C7	GANGJI MANISHA
9	14R11A05C8	GARIKAPATI NAMRATHA CHOWDARY
10	14R11A05C9	GARLAPATI RENUKA
11	14R11A05D0	GUNDARAPU SOUJANYA REDDY
12	14R11A05D1	GUNGI SAI KUMAR
13	14R11A05D2	GUNNALA RAMYA SREE
14	14R11A05D3	ILA BHAVANA
15	14R11A05D4	J HEMANTH SAI
16	14R11A05D5	JELLA MOUNICA
17	14R11A05D6	K TEJA ABHINAV
18	14R11A05D7	KANAKADANDI NAGA VENKATA APARNA
19	14R11A05D8	KASI SRAVYA
20	14R11A05D9	KAVYA S
21	14R11A05E0	KUKKALA DEEPIKA
22	14R11A05E1	LAHIRI KOTAMRAJU
23	14R11A05E2	MARAM RAJEEV KUMAR
24	14R11A05E3	MARGONWAR GAYATRI
25	14R11A05E4	MEKALA SWATHI
26	14R11A05E5	MOTHABOINA KAMESHWAR RAO ROHAN MUDH
27	14R11A05E6	NALABOTHULA HARIKA
28	14R11A05E7	NALADALA SAI CHARITHA
29	14R11A05E8	NEMURI SAI SAMPATH GOUD
30	14R11A05E9	NIKHIL THAPA
31	14R11A05F0	P ANUDEEP
32	14R11A05F1	P SAI PRASAD
33	14R11A05F2	PARIMI SAIESH
34	14R11A05F3	PINNINTI SAIKIRAN REDDY
35	14R11A05F4	POLAGOUNI VAISHNAVI GOUD
36	14R11A05F5	POTHURI VENKATA SAI PRIYANKA
37	14R11A05F6	RAHUL N D
38	14R11A05F7	RAI MEGHANA
39	14R11A05F8	RAYALA PRIYANKA
40	14R11A05F9	RAYAPROLU ASWINI
41	14R11A05G0	REEBA FATIMA
42	14R11A05G1	ROSHAN ROY

43	14R11A05G2	S HARI PRASAD
44	14R11A05G3	SAI PRIYA MALLIKA K
45	14R11A05G4	SANKARAMANCHI MOUNIKA
46	14R11A05G5	SEEMALA MAHENDER SUNNY SAMUEL
47	14R11A05G6	SUJAN MOHAN PILLI
48	14R11A05G7	T C KAVERI
49	14R11A05G8	T SWATHI
50	14R11A05G9	TERETIPALLY KARUNA SRI
51	14R11A05H0	V ABHISHEK
52	14R11A05H1	V VENKATA SAI YASASWI
53	14R11A05H2	VALLAP NITHIN REDDY
54	14R11A05H3	VASI SAI DINESH
55	14R11A05H4	VATTI BHARGAVI
56	14R11A05H5	VELPULA SANDHYA RANI
57	14R11A05H6	VIDYA SRIJA VOLETI
58	14R11A05H7	Y SREEJA
59	15R15A0503	MAALOTHU GANESH
60	15R18A0501	SANDEEP G BURUD

Total: 60 Males: 27 Females: 33

**Class / Section: CSE 22D**

1	14R11A05H9	A SAHITH REDDY
2	14R11A05J0	ADITYA V S P
3	14R11A05J1	ANEM PUNEET SURYA
4	14R11A05J2	ANNAVARAPU NAGA SAI SRILEKHA
5	14R11A05J4	B VINAY
6	14R11A05J5	BALABADRA SNEHA
7	14R11A05J6	BAYYAPU KEERTHANA
8	14R11A05J7	BHASWANTH K
9	14R11A05J8	BIJUMALLA SETHU MADHAV
10	14R11A05J9	BILLA HRIDAY VIKAS
11	14R11A05K0	BOLLEPALLY DIVYA
12	14R11A05K1	BUGATHA BALA GAYATRI
13	14R11A05K2	BYRU AKHILA
14	14R11A05K3	DHONEPUDI SHASHIKANTH
15	14R11A05K4	G DEEPIKA
16	14R11A05K5	GADDAM SUMIKENDAR REDDY
17	14R11A05K6	GARIKIPATI VINEETHA
18	14R11A05K7	GUNISSETTY AKHIL
19	14R11A05K8	J ALIVELUMANGAVATHI
20	14R11A05K9	JESSICA SHINY THOMAS
21	14R11A05L0	KANDI SAI JAGADISH

22	14R11A05L1	KARTHIK KALVAGADDA
23	14R11A05L2	KATARAY SAMYUKTHA DEVI
24	14R11A05L3	KATHI GANESH GOUD
25	14R11A05L4	KATKAM SAI SRI
26	14R11A05L5	KATKURI NIHARIKA
27	14R11A05L6	KODANDAM SAI PRATHYUSH REDDY
28	14R11A05L7	KONDOJU ABHISHEK KUMAR
29	14R11A05L8	KOONAPAREDDY ETHESH KUMAR
30	14R11A05L9	MADDURU VENKATA SAI SHRUTHI
31	14R11A05M0	MANDAVILLI LALITH KUMAR
32	14R11A05M1	MARGUM ALEKHYA
33	14R11A05M2	MD AKBAR
34	14R11A05M3	MEKA NAVEENA
35	14R11A05M4	MOHAMMED FURQUAN AHMED
36	14R11A05M5	MOUNIKA V
37	14R11A05M6	MUTHYAM LIKITHA SREE
38	14R11A05M7	NARU SIVA SAI KARTHIK
39	14R11A05M8	NEMANI HEMANTH KUMAR
40	14R11A05M9	NUTHALAKANTI PRANAV KUMAR NANU
41	14R11A05N0	P DIVYA TEJA
42	14R11A05N1	PANANGIPALLI NAGA SAI MOUNIKA
43	14R11A05N2	PATURU NAGA SRI NIKHILENDRA
44	14R11A05N3	PRIYANKA PANDA
45	14R11A05N4	PS MANIVENKAT RATNAM MUTYALA
46	14R11A05N5	R.HARI SHANKER REDDY
47	14R11A05N6	RAMYA DUBAGUNTA
48	14R11A05N7	SATLA MOUNIKA
49	14R11A05N8	SHAIK JAVEERIA
50	14R11A05N9	SINGIREDDY AKHILA
51	14R11A05P0	SINGIREDDY MANEESHA
52	14R11A05P1	SOMIDI LEKHANA
53	14R11A05P2	T HARSHINI
54	14R11A05P3	TALATH FATHIMA
55	14R11A05P4	THOUDOJU RAMAKRISHNA
56	14R11A05P5	TIRUMALARAJU KARTHIK RAMA KRISHNA V
57	14R11A05P6	VALABOJU VAMSHI KRISHNA
58	14R11A05P7	VARUN R SHAH
59	15R15A0504	ACHHI PHANINDRA
60	15R15A0505	P NANDA SAI
Total: 60 Males: 31 Females: 29		
Grand Total: 240( Males:109 Females:131)		

SINo	AdmnNo	StudentName	group
<b>Class / Section: CSE 22A</b>			
1	13R11A05G3	SRIKANTH AKKANAPALLY	I
2	14R11A0501	AKHILESH G	
3	14R11A0502	AKULA KRISHNA	
4	14R11A0503	ALLAM USHA SREE	
5	14R11A0504	ANANT SRIVATS	
6	14R11A0505	AYYAGARI V S V VARDHAMAN	II
7	14R11A0506	B MOHANA SREYA	
8	14R11A0507	B PRASHANTH	
9	14R11A0508	BANDA GAYATHRI VEENA	
10	14R11A0509	BATTAR SRIKANTH RAGHAVA	
11	14R11A0510	BHONSLE SAI KRUTHI	III
12	14R11A0511	BONAGIRI TEJASHWINI	
13	14R11A0512	BOTLA MOUNIKA	
14	14R11A0513	D VENKATESH	
15	14R11A0514	DASARI PRANAY KUMAR	
16	14R11A0515	DEEPIKA TUDIMILA	IV
17	14R11A0516	DOKKA DIVYA	
18	14R11A0517	E MADHU BHARGAVA	
19	14R11A0518	K JWALA	
20	14R11A0519	K SAMHITHA REDDY	
21	14R11A0520	K THAPASVI REDDY	V
22	14R11A0521	KALAKONDA VAISHNAVI	
23	14R11A0522	KAMPALLY YAMINI	
24	14R11A0523	KONTHAM RAVITEJA REDDY	
25	14R11A0524	KOSARAJU LEELA KRISHNA	
26	14R11A0525	KRISHNA SWETHA R	VI
27	14R11A0526	KUNDUR ROHAN REDDY	
28	14R11A0527	KUNNUMAL MAVILA AMRITHA	
29	14R11A0528	KURMA BHARGAV	
30	14R11A0529	KUTHURU PRIYANKA	
31	14R11A0530	KYASNOORI SHIVANI	VII
32	14R11A0531	M K SREE HARSHA	
33	14R11A0532	M POOJA	
34	14R11A0533	M SAI KRISHNA	
35	14R11A0534	M SHANMUGHA PRIYA	
36	14R11A0535	MALLADI RAMYA	VIII
37	14R11A0536	MEESALA SHAILAJA	
38	14R11A0537	N SRI MANASVI	
39	14R11A0538	NADENDLA VENKATA VINEET MURARI	
40	14R11A0539	NAKKA AKHIL	
41	14R11A0540	NALLANAGULA BHANU PRAKASH	IX

42	14R11A0541	P SOWMYA	
43	14R11A0542	PARVATHALA CHETANA	
44	14R11A0543	PENDYALA HEMANTH SAI	
45	14R11A0544	R CHANDRA MOHAN	
46	14R11A0545	REKULA CHANDRAHASA	X
47	14R11A0546	RUDRARAJU PAVANI	
48	14R11A0547	S SWATHI	
49	14R11A0548	SAI SINDHU BEERAM	
50	14R11A0549	SOPPADANDI AISHWARYA RAJ	
51	14R11A0550	SYED VAQUAS ASHRAF	XI
52	14R11A0551	TALLURI MURALI KRISHNA	
53	14R11A0552	TAMMEWAR SNEHIT	
54	14R11A0553	THIRIVEEDI DHEERAJ KUMAR	
55	14R11A0554	THUMMA VINEETHA REDDY	
56	14R11A0555	VALLURU VENUMADHURI	XII
57	14R11A0556	VAMANA GUNTLA MANOJ	
58	14R11A0557	VENKATA SAITEJA Y	
59	14R11A0558	VUSAKA VIKAS REDDY	
60	14R11A0559	Y JYOTSNA	
Total: 60 Males: 29 Females: 31			
<b>Class / Section: CSE 22B</b>			
1	13R11A0561	BATTU SHIRISH KUMAR REDDY	I
2	14R11A0560	AKELLA SUBRAMANYA SOUJANYA	
3	14R11A0561	ALEENA CHACKO K	
4	14R11A0562	BARLA PRAVALIKA	
5	14R11A0563	BATHINI DIVYA	
6	14R11A0564	BOLLA NAVEEN REDDY	II
7	14R11A0565	CHAVALI SWATHI	
8	14R11A0566	CHEPYALA AKHIL	
9	14R11A0567	CHITTARVU LAKSHMI ISHWARYA	
10	14R11A0568	DAMAGALLA KARTIK	
11	14R11A0569	DHATRI NAIDU BHOGADI	III
12	14R11A0571	DODDAPANENI SANOJ	
13	14R11A0572	DUGGISHETTY SAI PRASANNA	
14	14R11A0573	E PAVAN CHANDRA	
15	14R11A0574	G BHAVANA	
16	14R11A0575	G KISHAN	IV
17	14R11A0576	G LASYA PRIYA	
18	14R11A0577	G SIRIVENNELA	
19	14R11A0578	GANAPATHI RAJU MADHURI	
20	14R11A0579	GANGAVARAPU MANASA	

21	14R11A0580	GATTOJI HARITHA	V
22	14R11A0581	GAYATHRI MANDAL	
23	14R11A0582	GORTI SANTOSH KUMAR	
24	14R11A0583	HRITIK S	
25	14R11A0584	KALYANAM KRUTHIKA REDDY	VI
26	14R11A0585	KANUMURI SRI PRATHYUSHA	
27	14R11A0586	KOLLU VINESH BABU	
28	14R11A0587	KONDISETTI NIRANJANA KUMARI	
29	14R11A0588	KONIKA VENKATA PADMA PRIYA HASINI	VII
30	14R11A0589	KURUP MEGHANA	
31	14R11A0590	MADARAJU RAMYA SAI	
32	14R11A0591	MEDIPALLY SRIYA	
33	14R11A0592	MUDDU DEEPTHI	VIII
34	14R11A0593	N LAHARI	
35	14R11A0594	N VIKHYAT	
36	14R11A0595	NAKEERTHA SANDHYA RANI	
37	14R11A0596	NALLAMILLI JYOTHI	IX
38	14R11A0597	NALLAPANENI ADITYA SAI	
39	14R11A0598	NAMBURI MANISHA	
40	14R11A0599	NIKHIL SINGH P	
41	14R11A05A0	ORUGANTI KALPANA	X
42	14R11A05A1	PATLORI SAI KUMAR	
43	14R11A05A2	PEESAPATI VENKATA SAI VAMSI	
44	14R11A05A3	PINDIPOLU SRAVYA	
45	14R11A05A4	PRANATHI KRISHNA SANGANI	XI
46	14R11A05A5	PUDHOTA AVINASH	
47	14R11A05A6	S ASHA LAKSHMI	
48	14R11A05A7	S CHANDANA REDDY	
49	14R11A05A8	SHAIK HAFEEZ HUSSAIN	XII
50	14R11A05A9	SHEELAM SUSHMA	
51	14R11A05B0	SRAVIKA SANKU	
52	14R11A05B1	THUMMALA RISHIKA REDDY	
53	14R11A05B2	VARSHA CHAHAL	XII
54	14R11A05B3	VEERAVALLI SHILPA	
55	14R11A05B4	VUPPUTTURI SUSHANTH KUMAR	
56	14R11A05B5	Y ADITHYA	
57	14R11A05B6	YEDDU SHASHI KUMAR	XII
58	14R11A05B7	KATAMONI HARSHITHA	
59	15R15A0501	B VINESH	
60	15R15A0502	B SHRAVAN KUMAR	
Total: 60 Males: 22 Females: 38			

Class / Section: CSE 22C			
1	14R11A05C0	A S SRUJAN SAI	I
2	14R11A05C1	AITHA RANJEETH KUMAR	
3	14R11A05C2	ALIGETI MAHITHA	
4	14R11A05C3	BAGGU VISHNU SAI PRASAD	
5	14R11A05C4	BARGELA PRANAY RAJ	
6	14R11A05C5	BIROJU SAI SUGANDH CHARY	II
7	14R11A05C6	G AKHILA	
8	14R11A05C7	GANGJI MANISHA	
9	14R11A05C8	GARIKAPATI NAMRATHA CHOWDARY	
10	14R11A05C9	GARLAPATI RENUKA	
11	14R11A05D0	GUNDARAPU SOUJANYA REDDY	III
12	14R11A05D1	GUNGI SAI KUMAR	
13	14R11A05D2	GUNNALA RAMYA SREE	
14	14R11A05D3	ILA BHAVANA	
15	14R11A05D4	J HEMANTH SAI	
16	14R11A05D5	JELLA MOUNICA	IV
17	14R11A05D6	K TEJA ABHINAV	
18	14R11A05D7	KANAKADANDI NAGA VENKATA APARNA	
19	14R11A05D8	KASI SRAVYA	
20	14R11A05D9	KAVYA S	
21	14R11A05E0	KUKKALA DEEPIKA	V
22	14R11A05E1	LAHIRI KOTAMRAJU	
23	14R11A05E2	MARAM RAJEEV KUMAR	
24	14R11A05E3	MARGONWAR GAYATRI	
25	14R11A05E4	MEKALA SWATHI	
26	14R11A05E5	MOTHABOINA KAMESHWAR RAO ROHAN MUDH	VI
27	14R11A05E6	NALABOTHULA HARIKA	
28	14R11A05E7	NALADALA SAI CHARITHA	
29	14R11A05E8	NEMURI SAI SAMPATH GOUD	
30	14R11A05E9	NIKHIL THAPA	
31	14R11A05F0	P ANUDEEP	VII
32	14R11A05F1	P SAI PRASAD	
33	14R11A05F2	PARIMI SAIESH	
34	14R11A05F3	PINNINTI SAIKIRAN REDDY	
35	14R11A05F4	POLAGOUNI VAISHNAVI GOUD	
36	14R11A05F5	POTHURI VENKATA SAI PRIYANKA	VIII
37	14R11A05F6	RAHUL N D	
38	14R11A05F7	RAI MEGHANA	
39	14R11A05F8	RAYALA PRIYANKA	
40	14R11A05F9	RAYAPROLU ASWINI	
41	14R11A05G0	REEBA FATIMA	IX
42	14R11A05G1	ROSHAN ROY	

43	14R11A05G2	S HARI PRASAD	
44	14R11A05G3	SAI PRIYA MALLIKA K	
45	14R11A05G4	SANKARAMANCHI MOUNIKA	
46	14R11A05G5	SEEMALA MAHENDER SUNNY SAMUEL	X
47	14R11A05G6	SUJAN MOHAN PILLI	
48	14R11A05G7	T C KAVERI	
49	14R11A05G8	T SWATHI	
50	14R11A05G9	TERETIPALLY KARUNA SRI	
51	14R11A05H0	V ABHISHEK	XI
52	14R11A05H1	V VENKATA SAI YASASWI	
53	14R11A05H2	VALLAP NITHIN REDDY	
54	14R11A05H3	VASI SAI DINESH	
55	14R11A05H4	VATTI BHARGAVI	
56	14R11A05H5	VELPULA SANDHYA RANI	XII
57	14R11A05H6	VIDYA SRIJA VOLETI	
58	14R11A05H7	Y SREEJA	
59	15R15A0503	MAALOTHU GANESH	
60	15R18A0501	SANDEEP G BURUD	
Total: 60 Males: 27 Females: 33			
<b>Class / Section: CSE 22D</b>			
1	14R11A05H9	A SAHITH REDDY	I
2	14R11A05J0	ADITYA V S P	
3	14R11A05J1	ANEM PUNEET SURYA	
4	14R11A05J2	ANNAVARAPU NAGA SAI SRILEKHA	
5	14R11A05J4	B VINAY	
6	14R11A05J5	BALABADRA SNEHA	II
7	14R11A05J6	BAYYAPU KEERTHANA	
8	14R11A05J7	BHASWANTH K	
9	14R11A05J8	BIJUMALLA SETHU MADHAV	
10	14R11A05J9	BILLA HRIDAY VIKAS	
11	14R11A05K0	BOLLEPALLY DIVYA	III
12	14R11A05K1	BUGATHA BALA GAYATRI	
13	14R11A05K2	BYRU AKHILA	
14	14R11A05K3	DHONEPUDI SHASHIKANTH	
15	14R11A05K4	G DEEPIKA	
16	14R11A05K5	GADDAM SUMIKENDAR REDDY	IV
17	14R11A05K6	GARIKIPATI VINEETHA	
18	14R11A05K7	GUNISSETTY AKHIL	
19	14R11A05K8	J ALIVELUMANGAVATHI	
20	14R11A05K9	JESSICA SHINY THOMAS	
21	14R11A05L0	KANDI SAI JAGADISH	V

22	14R11A05L1	KARTHIK KALVAGADDA	
23	14R11A05L2	KATARAY SAMYUKTHA DEVI	
24	14R11A05L3	KATHI GANESH GOUD	
25	14R11A05L4	KATKAM SAI SRI	
26	14R11A05L5	KATKURI NIHARIKA	VI
27	14R11A05L6	KODANDAM SAI PRATHYUSH REDDY	
28	14R11A05L7	KONDOJU ABHISHEK KUMAR	
29	14R11A05L8	KOONAPAREDDY ETHESH KUMAR	
30	14R11A05L9	MADDURU VENKATA SAI SHRUTHI	
31	14R11A05M0	MANDAVILLI LALITH KUMAR	VII
32	14R11A05M1	MARGUM ALEKHYA	
33	14R11A05M2	MD AKBAR	
34	14R11A05M3	MEKA NAVEENA	
35	14R11A05M4	MOHAMMED FURQUAN AHMED	
36	14R11A05M5	MOUNIKA V	VIII
37	14R11A05M6	MUTHYAM LIKITHA SREE	
38	14R11A05M7	NARU SIVA SAI KARTHIK	
39	14R11A05M8	NEMANI HEMANTH KUMAR	
40	14R11A05M9	NUTHALAKANTI PRANAV KUMAR NANU	
41	14R11A05N0	P DIVYA TEJA	IX
42	14R11A05N1	PANANGIPALLI NAGA SAI MOUNIKA	
43	14R11A05N2	PATURU NAGA SRI NIKHILENDRA	
44	14R11A05N3	PRIYANKA PANDA	
45	14R11A05N4	PS MANIVENKAT RATNAM MUTYALA	
46	14R11A05N5	R.HARI SHANKER REDDY	X
47	14R11A05N6	RAMYA DUBAGUNTA	
48	14R11A05N7	SATLA MOUNIKA	
49	14R11A05N8	SHAIK JAVEERIA	
50	14R11A05N9	SINGIREDDY AKHILA	
51	14R11A05P0	SINGIREDDY MANEESHA	XI
52	14R11A05P1	SOMIDI LEKHANA	
53	14R11A05P2	T HARSHINI	
54	14R11A05P3	TALATH FATHIMA	
55	14R11A05P4	THOUDOJU RAMAKRISHNA	
56	14R11A05P5	TIRUMALARAJU KARTHIK RAMA KRISHNA V	XII
57	14R11A05P6	VALABOJU VAMSHI KRISHNA	
58	14R11A05P7	VARUN R SHAH	
59	15R15A0504	ACHHI PHANINDRA	
60	15R15A0505	P NANDA SAI	
Total: 60 Males: 31 Females: 29			
Grand Total: 240( Males:109 Females:131)			