2 Marks Questions

1. **Define class and object**

Ans - Class: A class is a user defined data type which groups data members and its associated functions together.

Object: It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods

2. **List any eight features of Java. (summer 19,)**

Ans - Features of Java:
1. Data Abstraction and Encapsulation
2. Inheritance
3. Polymorphism
4. Platform independence
5. Portability
6. Robust
7. Supports multithreading
8. Supports distributed applications
9. Secure
10. Architectural neutral
11. Dynamic

3. **Enlist any two bitwise operators (winter 23)**

Ans - Bitwise AND (&)

It is a binary operator denoted by the symbol **&**. It returns 1 if and only if both bits are 1, else returns 0.

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

It is a binary operator denoted by the symbol ^ (pronounced as caret). It returns 0 if both bits are the same, else returns 1

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

4. Name the wrapper class methods for the following:

(i) To convert string objects to primitive int.

(ii) To convert primitive int to string objects. (Summer 19)

Ans - (i) To convert string objects to primitive int:

String str=”5”;

int value = Integer.parseInt(str);

(ii) To convert primitive int to string objects:

int value=5;

String str=Integer.toString(value);

5. Define Constructor. List its types. (winter 19,winter 22)

Ans - Constructor: A constructor is a special member which initializes an object immediately upon creation. It has the same name as class name in which it resides and it is syntactically similar to any method. When a constructor is not defined, java executes a default constructor which initializes all numeric members to zero and other types to null or spaces. Once defined, constructor is automatically called immediately after the object is created before new operator completes.

Types of constructors:

1. Default constructor

2. Parameterized constructor

3. Copy constructor

6. Enlist access specifiers in Java (winter 22,Summer 23)

Ans - The access specifiers in java specify accessibility (scope) of a data member, method, constructor or class. There are 5 types of java access specifier:

- public
- private
- default (Friendly)
- protected

7. Define array. List its types. (Winter 19)

Ans - An array is a homogeneous data type where it can hold only objects of one data type.

Types of Array:
1)One-Dimensional

2)Two-Dimensional

8. Give use of garbage collection in java?(summer 23)

Ans - Garbage collection in Java is the automated process of deleting code that's no longer needed or used. This automatically frees up memory space and ideally makes coding Java apps easier for developers. Java applications are compiled into bytecode that may be executed by a JVM.

9. Draw structural diagram of fiber optic cable and write its functions.

Ans –

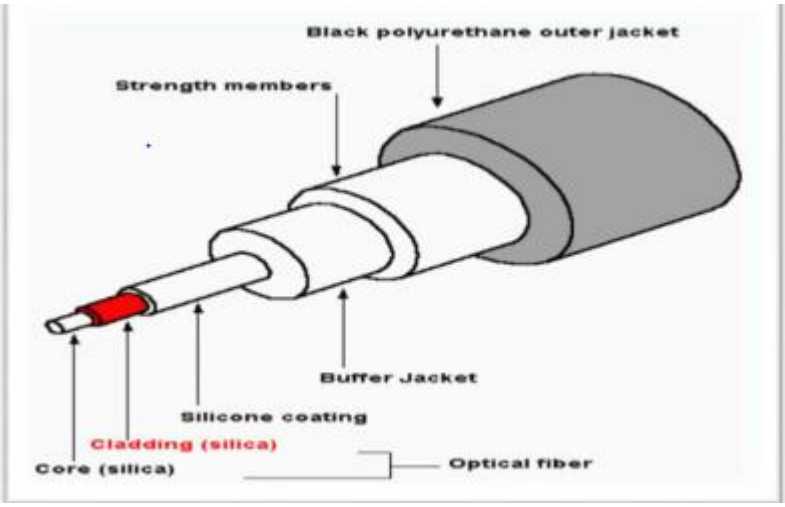


Fig. Structural diagram for Fibre Optic Cable

Functions of Optical Cable:

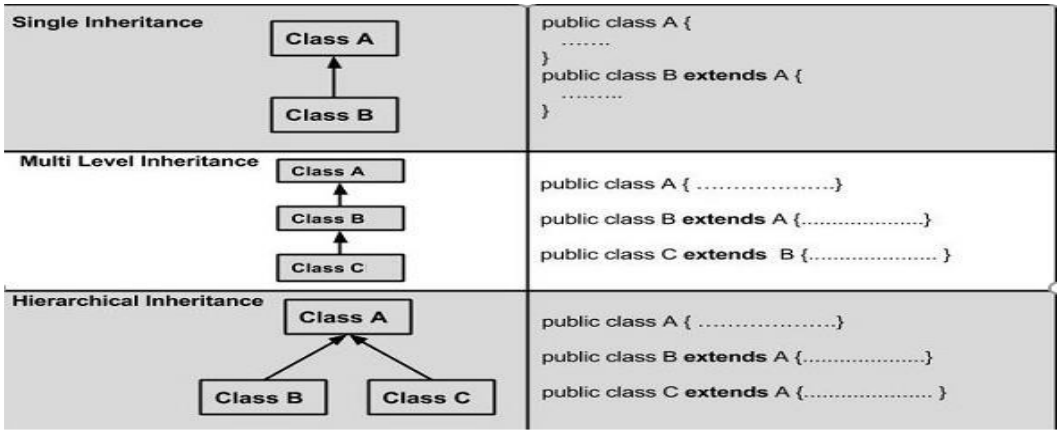
1. Single-mode fibers - Used to transmit one signal per fiber (used in telephones

and cable TV)

2. Multi-mode fibers - Used to transmit many signals per fiber (used in computer networks, local area networks)

10. List the types of inheritance which is supported by java. (Summer 19)(Winter 22)

Ans –



11. List any four Java API packages

Ans –

1.java.lang

2.java.util

3.java.io

4.java.awt

5.java.net

6.ava.applet

12. State use of finalize( ) method with its syntax.(Summer 19)

Ans –

Use of finalize( ):

Sometimes an object will need to perform some action when it is destroyed. Eg. If an object holding some non java resources such as

file handle or window character font, then before the object is

garbage collected these resources should be freed. To handle such

situations java provide a mechanism called finalization. In

finalization, specific actions that are to be done when an object is

garbage collected can be defined. To add finalizer to a class define

the finalize() method. The java run-time calls this method whenever it

is about to recycle an object.

Syntax:

protected void finalize() {

}

Notes By – Rajan Shukla

13. Give syntax to create a package and accessing package in java(Summer 23)

Ans - First create a folder and give name of package to it .

e.g.c:\mypack

for accessing package in java

import mypack.*;

to compile and run program

-d c:\mypack className.java

To run

Java className

14. List out different ways to access package from another package

Ans –

There are three ways to access the package from outside the package.

1) import package.*;

2) import package.classname;

15. Define error. List types of error.

Ans - Errors are mistakes that can make a program go wrong. Errors may be logical or may be typing mistakes. An error may produce an incorrect output or may terminate the execution of the program abruptly or even may cause the system to crash. Errors are broadly classified into two categories:

1. Compile time errors

2. Runtime errors

16. Write the syntax of try-catch-finally blocks.

Ans –

try{

//Statements to be monitored for any exception

} catch(ThrowableInstance1 obj) {

//Statements to execute if this type of exception occurs

} catch(ThrowableInstance2 obj2) {

//Statements

}finally{

//Statements which should be executed even if any exception happens

}

17. Define thread. Mention 2 ways to create thread.

Ans - 1. Thread is a smallest unit of executable code or a single task is also called as thread.

2. Each tread has its own local variable, program counter and lifetime.

3. A thread is similar to program that has a single flow of control.

18. Enlist any four compile time errors.

Ans –

1. Java source file name mismatch

2. Improper casing

3. Mismatched brackets

4. Missing semicolons

5. Method is undefined

6. Variable already defined

7. Variable not initialized

8. Type mismatch: cannot convert

9. Return type required

10. Unreachable code

19. Differentiate between starting thread with run() method and start()method

Ans - start method of thread class is implemented as when it is called a new Thread is created and code inside run() method is executed in that new Thread. While if run method is executed directly than no new Thread is created and code inside run() will execute on current Thread and no multi-threading will take place.

4 Marks Questions

1. State any four relational operators and their use(Winter 22)

Ans –

Operator	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
= =	Equal to
!=	Not equal to

2. Write a program to check whether the given number is prime or not.
Ans –

Code:

```
class PrimeExample
{
public static void main(String args[]){
int i,m=0,flag=0;
int n=7;//it is the number to be checked
m=n/2;
if(n==0||n==1){
System.out.println(n+" is not prime number");
}else{
for(i=2;i<=m;i++){
if(n%i==0){
System.out.println(n+" is not prime number");
flag=1;
break;
}
}
if(flag==0) { System.out.println(n+" is prime number"); }
}//end of else
}
}
```

Output:
7 is prime number

3. Explain any two logical operators in Java with example (winter 19,summer 22,winter 23)

Ans - Logical Operators: Logical operators are used when we want to form compound conditions by combining two or more relations. Java has three logical operators as shown in table

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

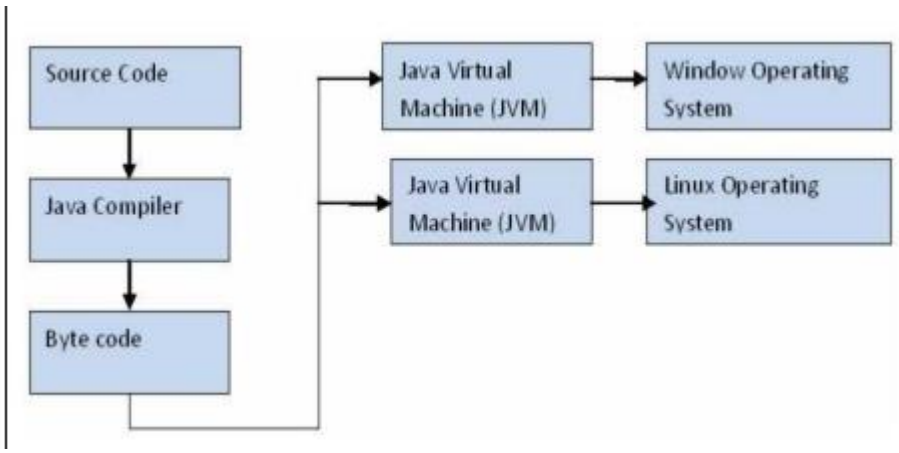
Program demonstrating logical Operators

```
public class Test
{
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 && b) = " + !(a && b));
 } }
```

Output: a && b = false
a || b = true
!(a && b) = true

4. Explain the concept of platform independence and portability with respect to Java language? (Summer 19)

Ans - Java is a platform independent language. This is possible because when a java program is compiled, an intermediate code called the byte code is obtained rather than the machine code.
Byte code is a highly optimized set of instructions designed to be executed by the JVM which is the interpreter for the byte code. Byte code is not a machine specific code. Byte code is a universal code and can be moved anywhere to any platform.
Therefore java is portable, as it can be carried to any platform. JVM is a virtual machine which exists inside the computer memory and is a simulated computer within a computer which does all the functions of a computer. Only the JVM needs to be implemented for each platform. Although the details of the JVM will defer from platform to platform, all interpret the same byte code.



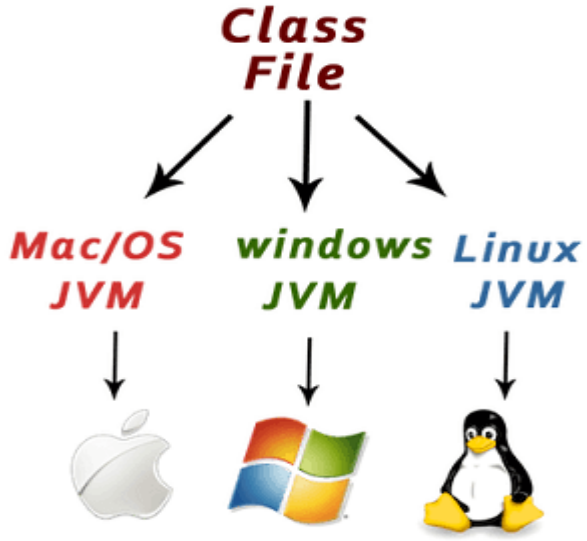
5. Write a program to find reverse of a number.(Winter 22)

Ans –

```
public class ReverseNumberExample1 {
public static void main(String[] args) {
 int number = 987654, reverse =0;
while(number !=0) {
int remainder = number % 10;
reverse = reverse * 10 + remainder;
number = number/10; }
System.out.printtln(“The reverse of the given number is: “ + reverse); } }
```

6. Explain any four features of Java. (Summer 22)

Ans - 1)Platform Independent
Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.
There are two types of platforms software-based and hardware-based. Java provides a software-based platform.



2)Robust
The English mining of Robust is strong. Java is robust because:
It uses strong memory management.
There is a lack of pointers that avoids security problems.
Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
There are exception handling and the type checking mechanism in Java. All these points make Java robust.
3)Architecture-neutral
Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

4)Simple
Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:
Java syntax is based on C++ (so easier for programmers to learn it after C++).
Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

7. Write all primitive data types available in Java with their storage Sizes in bytes. (winter 22)

Ans –

Data Type	Size
Byte	1 Byte
Short	2 Byte
Int	4 Byte
Long	8 Byte
Double	8 Byte
Float	4 Byte
Char	2 Byte
boolean	**1 Bit**

8. Write a Java program to find out the even numbers from 1 to 100 using for loop. (summer 22)

Ans -
```
public class DisplayEvenNumbersExample  {
public static void main(String args[])   {
int number=100;
System.out.print("List of even numbers from 1 to "+number+": ");
for (int i=1; i<=number; i++)   {
//logic to check if the number is even or not
//if i%2 is equal to zero, the number is even
if (i%2==0)   {
System.out.print(i + " ");
}  }  }  }
```
Output: List of even numbers from 1 to 100: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100

9. Explain switch case and conditional operator in java with suitable example(summer 22)

Ans –

The Java *switch statement* executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long.
There can be one or N number of case values for a switch expression.

The case value must be of switch expression type only. The case value must be literal or constant. It doesn't allow variables.
The case values must be unique. In case of duplicate value, it renders compile-time error.
The Java switch expression must be of byte, short, int, long (with its Wrapper type), enums and string.
Each case statement can have a break statement which is optional. When control reaches to the break statement, it jumps the control after the switch expression. If a break statement is not found, it executes the next case.
The case value can have a default label which is optional.
Syntax:

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
.....
    default:
  code to be executed if all cases are not matched;  }
```

Program for Switch case:

```
public class SwitchExample {
public static void main(String[] args) {
    //Declaring a variable for switch expression
    int number=20;
    //Switch expression
    switch(number){
    //Case statements
    case 10: System.out.println("10");
    break;
    case 20: System.out.println("20");
    break;
    case 30: System.out.println("30");
    break;
    //Default case statement
    default:System.out.println("Not in 10, 20 or 30");
    }
}
```

Output:

```
20
```

conditional operator ?:
The conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide; which value should be assigned to the variable. The operator is written as:
variable x = (expression)? value if true: value if false
Program:

```
public class Test {
   public static void main(String args[]) {
      int a, b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println("Value of b is: " + b);
      b = (a == 10) ? 20: 30;
      System.out.println(“Value of b is: " + b);
   }
}
```

Output

Value of b is: 30
Value of b is: 20

10. Write a program which display functioning of ATM machine(Hint:Withdraw,Deposite,Check Balance and Exit) (Winter 23)

Ans –

```
import java.io.*;

public class GFG {

 // Display current balance in account

   public static void displayBalance(int balance)  {

      System.out.println("Current Balance : " + balance);

      System.out.println(); }

  // Withdraw amount and update the balance

   public static int amountWithdrawing(int balance,  int withdrawAmount)

   {  System.out.println("Withdrawn Operation:");

      System.out.println("Withdrawing Amount :  + withdrawAmount);

      if (balance >= withdrawAmount) {

         balance = balance - withdrawAmount;

         System.out.println("Please collect your money and collect the card");

         displayBalance(balance);

      }

      else {

         System.out.println("Sorry! Insufficient Funds");

         System.out.println();  }
```

```
        return balance;
    } // Deposit amount and update the balance
    public static int amountDepositing(int balance, int depositAmount)
    {
        System.out.println("Deposit Operation:");
        System.out.println("Depositing Amount : " + depositAmount);
        balance = balance + depositAmount;
        System.out.println(  "Your Money has been successfully deposited");
        displayBalance(balance);
        return balance; }
  public static void main(String args[])  {
        int balance = 10000;
        int withdrawAmount = 5000;
int depositAmount = 2000;


        // calling display balance
        displayBalance(balance);
        // withdrawing amount
        balance  = amountWithdrawing(balance, withdrawAmount);
        // depositing amount
        balance = amountDepositing(balance, depositAmount);
```

}}

Output

Current Balance : 10000

Withdrawn Operation:

Withdrawing Amount : 5000

Please collect your money and collect the card

Current Balance : 5000

Deposit Operation:

Depositing Amount : 2000

Your Money has been successfully deposited

Balance : 7000

11. Describe instance Of and dot (.) operators in Java with suitable example.(summer 19)

Ans - Instance of operator:

The java instance of operator is used to test whether the object is an instance of the specified type (class or subclass or interface).

The instance of in java is also known as type cmparison operator because it compares the instance with type. It returns either true or false. If we apply the instance of operator with any variable that has null value, it returns false.

Example

```
class Simple1{
public static void main(String args[]){
Simple1 s=new Simple1();
System.out.println(sinstanceofSimple1);//true
}
}
```

dot (.) operator:

The dot operator, also known as separator or period used to separate a variable or method from a reference variable. Only static variables or methods can be accessed using class name. Code that is outside the object's class must use an object reference or expression, followed by

the dot (.) operator, followed by a simple field name.

Example

this.name=”john”; where name is a instance variable referenced by

‘this’ keyword

c.getdata(); where getdata() is a method invoked on object ‘c’.

12. Explain the types of constructors in Java with suitable example. (summer 19,winter 19 ,Summer 23)

(Note: Any two types shall be considered)

Ans –

Constructors are used to initialize an object as soon as it is created.

Every time an object is created using the ‘new’ keyword, a

constructor is invoked. If no constructor is defined in a class, java

compiler creates a default constructor. Constructors are similar to

methods but with to differences, constructor has the same name as

that of the class and it does not return any value.

The types of constructors are:

1. Default constructor

2. Constructor with no arguments

3. Parameterized constructor

4. Copy constructor

1. Default constructor: Java automatically creates default constructor

if there is no default or parameterized constructor written by user.

Default constructor in Java initializes member data variable to default

values (numeric values are initialized as 0, Boolean is initialized as

false and references are initialized as null).

```
class test1 {
int i;
boolean b;
byte bt;
```

```
float ft;
String s;
public static void main(String args[]) {
test1 t = new test1(); // default constructor is called.
System.out.println(t.i);
System.out.println(t.s);
System.out.println(t.b);
System.out.println(t.bt);
System.out.println(t.ft);
}
}
```

2. Constructor with no arguments: Such constructors does not have any parameters. All the objects created using this type of constructors has the same values for its datamembers.

Eg:

```
class Student {
int roll_no;
String name;
Student() {
roll_no = 50;
name="ABC";
}
void display() {
System.out.println("Roll no is: "+roll_no);
System.out.println("Name is : "+name);
}
public static void main(String a[]) {
Student s = new Student();
s.display();
}
}
```

3**. Parametrized constructor**: Such constructor consists of parameters. Such constructors can be used to create different objects with datamembers having different values.

```
class Student {
int roll_no;
String name;
Student(int r, String n) {
roll_no = r;
name=n;
}
void display() {
System.out.println("Roll no is: "+roll_no);
System.out.println("Name is : "+name);
}
public static void main(String a[]) {
Student s = new Student(20,"ABC");
s.display();
}
}
```

4. Copy Constructor : A copy constructor is a constructor that creates a new object using an existing object of the same class and initializes each instance variable of newly created object with corresponding instance variables of the existing object passed as argument. This constructor takes a single argument whose type is that of the class containing the constructor.

```
class Rectangle
{
int length;
int breadth;
Rectangle(int l, int b)
{
```

```
length = l;
breadth= b;
}
//copy constructor
Rectangle(Rectangle obj)
{
length = obj.length;
breadth= obj.breadth;
}
public static void main(String[] args)
{
Rectangle r1= new Rectangle(5,6);
Rectangle r2= new Rectangle(r1);
System.out.println("Area of First Rectangle : "+
(r1.length*r1.breadth));
System .out.println("Area of First Second Rectangle : "+
(r1.length*r1.breadth));
}
}
```

13. Define a class student with int id and string name as data members and a method void SetData ( ). Accept and display the data for five students. (summer 2019)

Ans –

```
import java.io.*;
class student
{
int id;
String name;
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
void SetData()
```

```
{
try
{
System.out.println("enter id and name for student");
id=Integer.parseInt(br.readLine());
name=br.readLine();
}
catch(Exception ex)
{}
}
void display()
{
System.out.println("The id is " + id + " and the name is "+ name);
}
public static void main(String are[])
{
student[] arr;
arr = new student[5];
int i;
for(i=0;i<5;i++)
{
arr[i] = new student();
}
for(i=0;i<5;i++)
{
arr[i].SetData();
}
for(i=0;i<5;i++)
{
arr[i].display();
}
```

}

}

14. Differentiate between String and String Buffer. (winter 19,Summer 22,Summer 23)

Ans –

String	String Buffer c
String is a major class	String Buffer is a peer class of String
Length is fixed (immutable)	Length is flexible (mutable)
Contents of object cannot be modified	Contents of object can be modified
Object can be created by assigning String constants enclosed in double quotes.	Objects can be created by calling constructor of String Buffer class using "new"
Ex:- String s="abc";	Ex:- StringBuffer s=new StringBuffer ("abc");

15. Differentiate between array and vector. (winter19,winter 22,summer 22)

Ans –

Array	Vector
1) An array is a structure that holds multiple values of the same type.	1)The Vector is similar to array holds multiple objects and like an array; it contains components that can be accessed using an integer index.

2) An array is a homogeneous data type where it can hold only objects of one data type.	2) Vectors are heterogeneous. You can have objects of different data types inside a Vector.
3) After creation, an array is a fixed-length structure.	3) The size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.
4) Array can store primitive type data element.	4) Vector are store non-primitive type data element.
5)Declaration of an array : int arr[] = new int [10];	5)Declaration of Vector: Vector list = new Vector(3);
6) Array is the static memory allocation.	6) Vector is the dynamic memory allocation.

16. List any four methods of string class and state the use of each. (Winter 19,Summer 22)

Ans - The java.lang.String class provides a lot of methods to work on

string. By the help of these methods,

We can perform operations on string such as trimming,

concatenating, converting, comparing, replacing strings etc.

1) to Lowercase (): Converts all of the characters in this String

to lower case.

Syntax: s1.toLowerCase()

Example: String s="Sachin";

System.out.println(s.toLowerCase());

Output: sachin

2)to Uppercase(): Converts all of the characters in this String to

upper case

Syntax: s1.toUpperCase()

Example:

String s="Sachin";

System.out.println(s.toUpperCase());

Output: SACHIN

3) trim (): Returns a copy of the string, with leading and trailing

whitespace omitted.

Syntax: s1.trim()

Example:

String s=" Sachin ";

System.out.println(s.trim());

Output:Sachin

4) replace (): Returns a new string resulting from replacing all

occurrences of old Char in this string with new Char.

Syntax: s1.replace('x','y')

Example:

String s1="Java is a programming language. Java is a platform.";

String s2=s1.replace("Java","Kava"); //replaces all occurrences

of "Java" to "Kava"

System.out.println(s2);

Output: Kava is a programming language. Kava is a platform

17. Define a class employee with data members 'empid , name and salary. Accept data for three objects and display it (winter 22)

Ans –

```
class employee
{
int empid;
String name;
double salary;
void getdata()
{
```

```
BufferedReader obj = new BufferedReader (new InputStreamReader(System.in));
System.out.print("Enter Emp number : ");
empid=Integer.parseInt(obj.readLine());
System.out.print("Enter Emp Name : ");
name=obj.readLine();
System.out.print("Enter Emp Salary : ");
salary=Double.parseDouble(obj.readLine());
}
void show()
{
System.out.println("Emp ID : " + empid);
System.out.println("Name : " + name);
System.out.println("Salary : " + salary);
}
}
classEmpDetails
{
public static void main(String args[])
{
employee e[] = new employee[3];
for(inti=0; i<3; i++)
{
e[i] = new employee(); e[i].getdata();
}
System.out.println(" Employee Details are : ");
for(inti=0; i<3; i++)
e[i].show();
}
}
```

18. Write a program to add 2 integer, 2 string and 2 float values is a vector. Remove the element specified by the user and display the list. (winter 22)

Ans –

```
import java.io.*;
import java.lang.*;
import java.util.*;
class vector2
{
public static void main(String args[])
{
vector v=new vector();
Integer s1=new Integer(1);
Integer s2=new Integer(2);
String s3=new String("fy");

String s4=new String("sy");
Float s7=new Float(1.1f);
Float s8=new Float(1.2f);
v.addElement(s1);
v.addElement(s2);
v.addElement(s3);
v.addElement(s4);
```

```
v.addElement(s7);
v.addElement(s8);
System.out.println(v);
v.removeElement(s2);
v.removeElementAt(4);
System.out.println(v);
}
}
```

19. Explain four methods of vector class with example? (Summer 23)

Ans –

1 Java Vector add() Method

The **add()** is a Java Vector class method which is used to insert the specified element in the given Vector. There are two different types of Java add() method which can be differentiated depending on its parameter. These are:

1.Java Vector add(int index, E element) Method

2.Java Vector add(E e) Method

Example:

```
import java.util.Vector;

public class VectorAddExample1 {

   public static void main(String args[]) {

      //Create an empty Vector with an initial capacity of 5

        Vector<String> vc = new Vector<>(4);

        //Add elements in the vector by using add() method

        vc.add("A");

        vc.add("B");

        vc.add("C");

        vc.add("D");

        vc.add("E");

        //Print all the elements of a Vector
```

```
        System.out.println("--Elements of Vector are--");
        for (String str : vc) {
          System.out.println("Alphabet= " +str);
        } } }
```

Output:

```
--Elements of Vector are--
Alphabet= A
Alphabet= B
Alphabet= C
Alphabet= D
Alphabet= E
```

2 Java Vector clear() Method

The **clear()** method of **Java Vector** class is used to remove all of the elements from the vector which is in use.

Syntax:

Following is the declaration of **clear()** method:

Public void clear()

Example:

```
import java.util.Vector;
  public class VectorClearExample1 {
    public static void main(String arg[]) {
      //Create an empty Vector
        Vector<String> vc = new Vector<>();
        //Add elements in the vector by using add() method
        vc.add("A");
        vc.add("B");
        vc.add("C");
```

```
        //Print the size of vector
        System.out.println("Size of Vector before clear() method: "+vc.size());
        //Clear the vector
        vc.clear();
        System.out.println("Size of Vector after clear() method: "+vc.size());
         }
    }
```

Output:

```
Size of Vector before clear() method: 3
Size of Vector after clear() method: 0
```

3 Java Vector get() Method

The **get()** method of Java Vector class is used to get the element at the specified position in the vector.

Syntax

Following is the declaration of **get()** method:

```
public E get(int index)
```

Example:

```
import java.util.*;
public class VectorGetExample1 {
   public static void main(String arg[]) {
      //Create an empty vector
      Vector<Integer> vec = new Vector<>();
      //Add element in the vector
      vec.add(1);
      vec.add(2);
      vec.add(3);
      vec.add(4);
```

```
        vec.add(5);

        //Get the element at specified index

        System.out.println("Element at index 1 is = "+vec.get(1));

        System.out.println("Element at index 3 is = "+vec.get(3));

        }

    }
```

Output:

```
Element at index 1 is = 2

Element at index 3 is = 4
```

4 Java Vector capacity() Method

The **capacity()** method of **Java Vector** class is used to get the current capacity of the vector which is in use.

Syntax:

Following is the declaration of **capacity()** method:

```
Public int capacity()
```

Example :

```
import java.util.Vector;

public class VectorCapacityExample1 {

   public static void main(String arg[]) {

      //Create an empty Vector with initial capacity of 5

      Vector<Integer> vecObject = new Vector<Integer>(5);

      //Add values in the vector

      vecObject.add(3);

      vecObject.add(5);

      vecObject.add(2);

      vecObject.add(4);
```

```
        vecObject.add(1);

         System.out.println("Current capacity of Vector is: "+vecObject.capacity());

      }
```

} **Output:**

Current capacity of Vector is: 5

20. What advantages does TDM have over FDM in a circuit switched network?

Ans –

- In TDM, each signal uses all of the bandwidth some of the time, while for FDM, each signal uses a small portion of the bandwidth all of the time.
- TDM uses the entire frequency range but dynamically allocates time, certain jobs might require less or more time, which TDM can offer but FDM is unable to as it cannot change the width of the allocated frequency.
- TDM provides much better flexibility compared to FDM.
- TDM offers efficient utilization of bandwidth
- Low interference of signal and minimizes cross talk

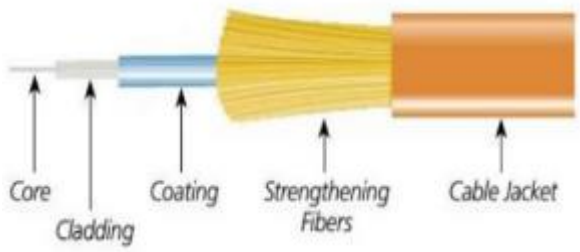
21. Differentiate between FDM and TDM

Ans –

Sr.No	FDM	TDM
1	FDM divides the channel into two or more frequency ranges that do not overlap.	TDM divides and allocates certain time periods to each channel in an alternating manner.
2	Frequency is shared.	Times scale is shared
3	Used with Analog signals	Used with both Digital signals and analog signals
4	Interference is high	Interference is Low or negligible
5	Utilization is Ineffective	Efficiently used

22. Draw and explain fiber optic cable.

Ans –



Notes By – Rajan Shukla

Fiber optic cable:

- A fiber-optic cable is made up of glass or plastic.
- It transmits signals in the form of light.
- The outer jacket is made up of PVC or Teflon.
- Kevlar strands are placed inside the jacket to strengthen the cable.
- Below the Kevlar strands, there is another plastic coating which acts as acushion.
- The fiber is at the center of the cable, and it consists of cladding and glass core.
- The density of the cladding is less than that of the core .

Optical fibers use the principle of 'reflection' to pass light through a channel.

23. State the two advantages and disadvantages of unguided media

Ans –

Advantages:

1 .Use for long distance communication.

2. High speed data transmission.

3. Many receiver stations can receive signals from same sender station

Disadvantages :

1..Radio waves travel through Lowest portion of atmosphere which can have lot of noise and interfering signals

2. Radio wave communication through unguided media is an insecure communication. 3.Radio wave propagation is susceptible to weather effects like rain, thunder and storm etc.

24. Describe Multiplexing techniques

Ans –

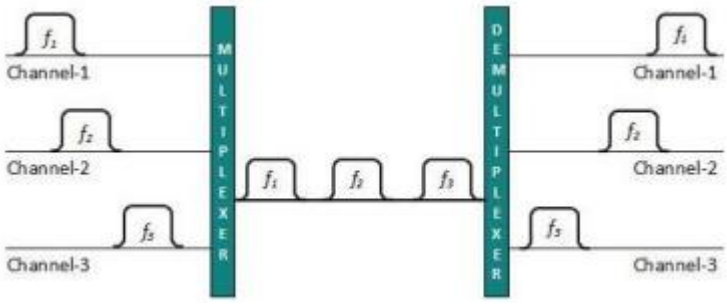
Multiplexing is a technique by which different analog and digital streams of transmission can be simultaneously processed over a shared link. Multiplexing divides the high capacity medium into low capacity logical medium which is then shared by different streams. Communication is possible over the air (radio frequency), using a physical media (cable), and light (optical fiber). All mediums are capable of multiplexing. When multiple senders try to send over a single medium, a device called Multiplexer divides the physical channel and allocates one to each. On the other end of communication, a De-multiplexer receives data from a single medium, identifies each, and sends to different receivers

Different multiplexing techniques are

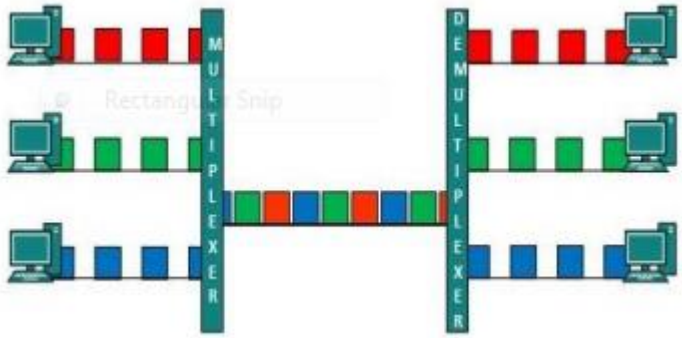
1.Frequency Division multiplexing

2.Time division multiplexing

Frequency Division Multiplexing: When the carrier is frequency, FDM is used. FDM is an analog technology. FDM divides the spectrum or carrier bandwidth in logical channels and allocates one user to each channel. Each user can use the channel frequency independently and has exclusive access of it. All channels are divided in such a way that they do not overlap with each other. Channels are separated by guard bands. Guard band is a frequency which is not used by either channel.



Time Division Multiplexing: TDM is applied primarily on digital signals but can be applied on analog signals as well. In TDM the shared channel is divided among its user by means of time slot. Each user can transmit data within the provided time slot only. Digital signals are divided in frames, equivalent to time slot i.e. frame of an optimal size which can be transmitted in given time slot. TDM works in synchronized mode. Both ends, i.e. Multiplexer and Demultiplexer are timely synchronized and both switch to next channel simultaneously.



When channel A transmits its frame at one end, the De-multiplexer provides media to channel A on the other end. As soon as the channel A's time slot expires, this side switches to channel B. On the other end, the De-multiplexer
works in a synchronized manner and provides media to channel B. Signals from different channels travel the path in interleaved manner.

25. Explain circuit switching networks with neat sketch.

Ans –

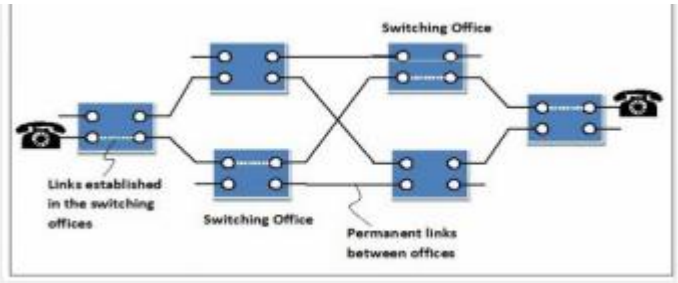
Circuit switching is a connection-oriented network switching technique. Here, a dedicated route is established between the source and the destination and the entire message is transferred through it

Phases of Circuit Switch Connection:

• Circuit Establishment: In this phase, a dedicated circuit is established from the source to the destination through a number of intermediate switching centers. The sender and receiver transmits communication signals to request and acknowledge establishment of circuits.

• Data Transfer: Once the circuit has been established, data and voice are transferred from the source to the destination. The dedicated connection remains as long as the end parties communicate.

• Circuit Disconnection: When data transfer is complete, the connection is relinquished. The disconnection is initiated by any one of the user. Disconnection involves removal of all intermediate links from the sender to the receiver.



The diagram represents circuit established between two telephones connected by circuit switched connection. The blue boxes represent the switching offices and their connection with other switching offices. The black lines connecting the switching offices represent the permanent link between the offices.

26. Draw a neat diagram of twisted pair cable and state its types.

Ans –

Fig.Twisted Pair cable

Twisted Pair Cables types :

1.Unshielded Twisted Pair Cables (UTP)
2.Shielded Twisted Pair Cables (STP)

27. Compare packet switched and circuit switched network

Ans –

Sr.No	Packet Switching	Circuit Switching
1	In Packet switching directly data transfer takes place.	In-circuit switching has there are 3 phases: i) Connection Establishment. ii) Data Transfer. iii) Connection Released.
2	In Packet switching, each data unit just knows the final destination address intermediate path is decided by the routers.	In-circuit switching, each data unit knows the entire path address which is provided by the source.
3	In Packet switching, data is processed at all intermediate nodes including the source system.	In-Circuit switching, data is processed at the source system only
4	The delay between data units in packet switching is not uniform.	The delay between data units in circuit switching is uniform.
5	Packet switching is less reliable.	Circuit switching is more reliable.
6	It is a store and forward technique.	It is not a store and forward technique.
7	No call setup is required in packet switching.	Call setup is required in circuit switching.
8	Packet switching is implemented at the datalink layer and network layer.	The circuit switching network is implemented at the physical layer.

28. State the use of final keyword with respect to inheritance. (Winter 22)

Ans –

Final keyword : The keyword final has three uses. First, it can be used to create the equivalent of a named constant.(in interface or class we use final as shared constant or constant.)

Other two uses of final apply to inheritance

Using final to Prevent Overriding While method overriding is one of Java's most powerful features, To disallow a method from being overridden, specify final as a modifier at the start of its declaration. Methods declared as final cannot be overridden.

The following fragment illustrates final:

```
class A
{
final void meth()
{
System.out.println("This is a final method.");

}
}
class B extends A
{
void meth()
{ // ERROR! Can't override.
System.out.println("Illegal!");
}
}
```

As base class declared method as a final , derived class can not override the definition of base class methods.

29. Differentiate between class and interfaces.(Winter 19)
Ans –

Class	Interface
1)doesn't Supports multiple inheritance	1) Supports multiple inheritance
2)"extend " keyword is used to inherit	2)"implements " keyword is used to inherit
3) class contain method body	3) interface contains abstract method(method without body)
4)contains any type of variable	4)contains only final variable
5)can have constructor	5)cannot have constructor
6)can have main() method	6)cannot have main() method

7)syntax
Class classname
{
Variable declaration,
Method declaration
}
	7)syntax
	Inteface Innterfacename
	{
	Final Variable declaration,
	abstract Method declaration
	}

30. Describe final variable and final method. (Winter 19)

Ans –

Final method: making a method final ensures that the

functionality defined in this method will never be altered in any

way, ie a final method cannot be overridden.

Syntax:

final void findAverage()

{

//implementation

}

Example of declaring a final method:

class A

{ final void show()

{

System.out.println(“in show of A”);

}

}

class B extends A

{

void show() // can not override because it is declared with final

{

System.out.println(“in show of B”);

}}

Final variable: the value of a final variable cannot be changed.

Final variable behaves like class variables and they do not take

any space on individual objects of the class.

Example of declaring final variable: final int size = 100;

31. Describe interface in Java with suitable example.

Ans –

The interface in Java is *a* mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritances in Java using Interface.

Syntax for Java Interfaces

interface {

  // declare constant fields

  // declare methods that abstract

  // by default.

}

```
import java.io.*;
// A simple interface
interface In1 {

   // public, static and final
   final int a = 10;

   // public and abstract
   void display();
}

// A class that implements the interface.
class TestClass implements In1 {

   // Implementing the capabilities of
   // interface.
   public void display(){
     System.out.println("Geek");
   }

   // Driver Code
   public static void main(String[] args)
   {
      TestClass t = new TestClass();
      t.display();
      System.out.println(a);
   }
}
```

Output

Geek

10

Notes By – Rajan Shukla

32. Write a program to show the Hierachical inheritance.

Ans –

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

Output:

meowing...

eating...

33. Explain dynamic method dispatch in Java with suitable example. .(Summer 19)6m & 4 m

Ans –

Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

 When an overridden method is called through a superclass reference, Java determines which version (superclass/subclasses) of that method is to be executed based upon the type of the object being referred to at the time the call occurs. Thus, this determination is made at run time.

 At run-time, it depends on the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed

 A superclass reference variable can refer to a subclass object. This is also known as upcasting. Java uses this fact to resolve calls to overridden methods at run time.

Therefore, if a superclass contains a method that is overridden by a subclass, then when different types of objects are referred to through a superclass reference variable, different versions of the method are executed. Here is an example that illustrates dynamic method dispatch:

```
// A Java program to illustrate Dynamic Method
// Dispatch using hierarchical inheritance
class A
{
void m1()
{
System.out.println("Inside A's m1 method");
}
}
class B extends A
{
// overriding m1()
void m1(){
System.out.println("Inside B's m1 method");
}
}
class C extends A
{
// overriding m1()
void m1()
```

```
{
System.out.println("Inside C's m1 method");
}
}
// Driver class
class Dispatch
{
public static void main(String args[])
{
// object of type A
A a = new A();
// object of type B
B b = new B();
// object of type C
C c = new C();
// obtain a reference of type A
A ref;
// ref refers to an A object
ref = a;
// calling A's version of m1()
ref.m1();
// now ref refers to a B object
ref = b;
// calling B's version of m1()
ref.m1();
// now ref refers to a C object
ref = c;
// calling C's version of m1()
ref.m1();
}
}
```

34. Define exception. State built-in exceptions.

Ans –

An exception is a problem that arises during the execution of a program. Java exception handling is used to handle error conditions in a program systematically by taking the necessary action

Built-in exceptions:

• Arithmetic exception: Arithmetic error such as division by zero.

• ArrayIndexOutOfBounds Exception: Array index is out of bound

• ClassNotFoundException

• FileNotFoundException: Caused by an attempt to access a nonexistent file.

• IO Exception: Caused by general I/O failures, such as inability to read from a file.

• NullPointerException: Caused by referencing a null object.

• NumberFormatException: Caused when a conversion between strings and number fails.

• StringIndexOutOfBoundsException: Caused when a program attempts to access a nonexistent character position in a string.

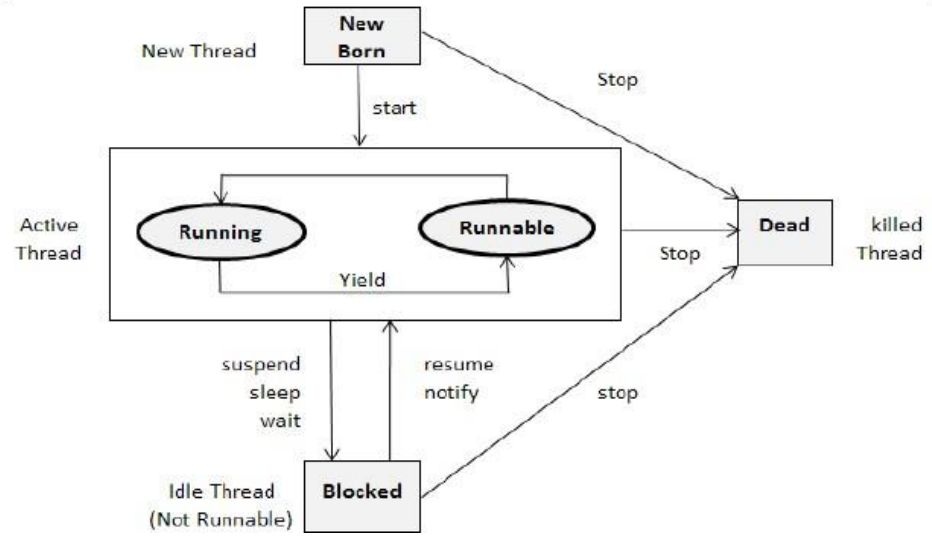
• OutOfMemoryException: Caused when there's not enough memory to allocate a new object.

• SecurityException: Caused when an applet tries to perform an action not allowed by the browser's security setting.

• StackOverflowException: Caused when the system runs out of stack space.

35. Explain life cycle of thread. (winter 22)(summer 19)(Summer 22)(winter 23)

Ans -



Thread Life Cycle Thread has five different states throughout its life.

1. Newborn State
2. Runnable State
3. Running State
4. Blocked State
5. Dead State

Thread should be in any one state of above and it can be move from one state toanother by different methods and ways.

Newborn state: When a thread object is created it is said to be in a new born state.When the thread is in a new born state it is not scheduled running from this state itcan be scheduled for running by start() or killed by stop(). If put in a queue it moves to runnable state.

Runnable State: It means that thread is ready for execution and is waiting for theavailability of the processor i.e. the thread has joined the queue and is waiting forexecution. If all threads have equal priority, then they are given time slots for execution in round robin fashion. The thread that relinquishes control joins
the queue at the end and again waits for its turn. A thread can relinquish thecontrol to another before its turn comes by yield().

Running State: It means that the processor has given its time to the thread for execution. The thread runs until it relinquishes control on its own or it is pre- empted by a higher priority thread.

Blocked state: A thread can be temporarily suspended or blocked from entering into the runnable and running state by using either of the following thread method.

1) **suspend()** : Thread can be suspended by this method. It can be rescheduled by resume().
2) **wait()**: If a thread requires to wait until some event occurs, it can be done usingwait method and can be scheduled to run again by notify().
3) **sleep():** We can put a thread to sleep for a specified time period using sleep(time) where time is in ms. It re-enters the runnable state as soon as periodhas elapsed /over

Dead State: Whenever we want to stop a thread form running further we can callits stop().The statement causes the thread to move to a dead state. A thread will
also move to dead state automatically when it reaches to end of the method. Thestop method may be used when the premature death is required.

36. Explain the two ways of creating threads in Java.

Ans –

Thread is a independent path of execution within a program.There are
two ways to create a thread:
1. By extending the Thread class.
Thread class provide constructors and methods to create and perform operationson a thread. This class implements the Runnable interface. When we extend the class Thread, we need to implement the method run(). Once we create an object,
we can call the start() of the thread class for executing the method run().

Eg:

```
class MyThread extends Thread {public
void run() {
for(int i = 1;i<=20;i++) {
```

```
System.out.println(i);
}
}
public static void main(String a[]) {
MyThread t = new MyThread(); t.start();
}
}
```

a. By implementing the runnable interface. Runnable interface has only on one method- run().Eg:

```
class MyThread implements Runnable {public
void run() {
for(int i = 1;i<=20;i++) {
System.out.println(i);
}
}
public static void main(String a[]) {
MyThread m = new MyThread(); Thread t =
new Thread(m);
t.start();
}}
```

37. Explain exception handling mechanism. w.r.t. try, catch, throw and finally.

Ans –

The **Exception Handling in Java** is one of the powerful *mechanism to handle theruntime errors* so that the normal flow of the application can be maintained. The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

Example :

```
class TestFinallyBlock {
public static void main(String args[]){
try{
//below code do not throw any exception
int data=25/5; System.out.println(data);
}
//catch won't be executed
catch(NullPointerException e){
. System.out.println(e);
. }
. //executed regardless of exception occurred or not
. finally {
. System.out.println("finally block is always executed");
```

```
. }
.
. System.out.println("rest of phe code...");
. }
. }
```

6 Marks Questions

1. Explain the command line arguments with suitable example (summer 19)

Ans –

Java Command Line Argument: The java command-line argument is an argument i.e. passed at the time of running the java program.
The arguments passed from the console can be received in the java program and it can be used as an input. So, it provides a convenient way to check the behaviour of the program for the different values.
You can pass N (1,2,3 and so on) numbers of arguments from the command prompt.
Command Line Arguments can be used to specify configuration information while launching your application.
There is no restriction on the number of java command line arguments.
You can specify any number of arguments Information is passed as Strings.
They are captured into the String args of your main method Simple example of command-line argument in java
In this example, we are receiving only one argument and printing it.
To run this java program, you must pass at least one argument from the command prompt.

```
class CommandLineExample {
public static void main(String args[]){
System.out.println("Your first argument is: "+args[0]);
} }
```

compile by > javac CommandLineExample.java
run by > java CommandLineExample sonoo

2. Describe the use of any methods of vector class with their syntax.

(Note: Any method other than this but in vector class shall be considered for answer). (summer 19)

Ans –

- boolean add(Object obj)-Appends the specified element to the end of this Vector.
- Boolean add(int index,Object obj)-Inserts the specified element at the specified position in this Vector.
- void addElement(Object obj)-Adds the specified component to

the end of this vector, increasing its size by one.

- int capacity()-Returns the current capacity of this vector.
- void clear()-Removes all of the elements from this vector.
- Object clone()-Returns a clone of this vector.
- boolean contains(Object elem)-Tests if the specified object is a component in this vector.
- void copyInto(Object[] anArray)-Copies the components of this vector into the specified array.
- Object firstElement()-Returns the first component (the item at index 0) of this vector.
- Object elementAt(int index)-Returns the component at the specified index.
- int indexOf(Object elem)-Searches for the first occurence of the given argument, testing for equality using the equals method.
- Object lastElement()-Returns the last component of the vector.
- Object insertElementAt(Object obj,int index)-Inserts the specified object as a component in this vector at the specified index.
- Object remove(int index)-Removes the element at the specified position in this vector.
- void removeAllElements()-Removes all components from this svector and sets its size to zero

3. Write a program to create a vector with five elements as (5,15, 25, 35, 45). Insert new element at 2nd position. Remove 1st and 4th element from vector. (winter 19)

Ans –

```
import java.util.*;

class VectorDemo

{

public static void main(String[] args)

{

Vector v = new Vector();

v.addElement(new Integer(5));

v.addElement(new Integer(15));

v.addElement(new Integer(25));

v.addElement(new Integer(35));

v.addElement(new Integer(45));
```

```
System.out.println("Original array elements are
");
for(int i=0;i<v.size();i++)
{
System.out.println(v.elementAt(i));
}
v.insertElementAt(new Integer(20),1); // insert
new element at 2nd position
v.removeElementAt(0);
//remove first element
v.removeElementAt(3);
//remove fourth element
System.out.println("Array elements after insert
and remove operation ");
for(int i=0;i<v.size();i++)
{
System.out.println(v.elementAt(i));
}}}
```

4. Write a program to check whether the string provided by the user is palindrome or not. (winter 22)

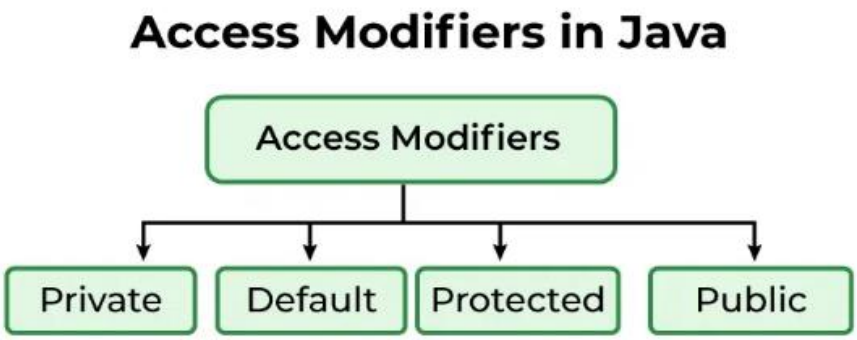
Ans –

```
import java.lang.*;
import java.io.*;
import java.util.*;
class palindrome
{
public static void main(String arg[ ]) throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter String:");
String word=br.readLine( );
int len=word.length( )-1;
int l=0;
int flag=1;
int r=len;
while(l<=r)
{
```

```
if(word.charAt(l)==word.charAt(r))
{
l++;
r--;
}
else
{
flag=0;
break;
}
}
if(flag==1)
{
System.out.println("palindrome");
}
else
{
System.out.println("not palindrome");
}
}
}
```

5. Explain any four visibility controls in Java.4m (summer 22)

Ans –



1. Default Access Modifier

When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default. The data members, classes, or methods that are not declared using any access modifiers i.e. having default access modifiers are accessible only within the same package.

```
package p1;


// Class Geek is having Defauslt access modifier
```

```
class Geek
{
   void display()
   {
      System.out.println("Hello World!");
   }
}
```

2. Private Access Modifier

The private access modifier is specified using the keyword **private**. The methods or data members declared as private are accessible only **within the class** in which they are declared.

```
package p1;


class A
{
private void display()
   {
      System.out.println("GeeksforGeeks");
   }
}
class B
{
public static void main(String args[])
   {
      A obj = new A();
      // Trying to access private method
      // of another class
      obj.display();
   }
```

}

3. Protected Access Modifier

The protected access modifier is specified using the keyword **protected.**

The methods or data members declared as protected are **accessible within the same package or subclasses in different packages.**

```
package p1;


// Class A

public class A

{

protected void display()

   {

      System.out.println("GeeksforGeeks");

   }

}

package p2;

import p1.*; // importing all classes in package p1


// Class B is subclass of A

class B extends A

{

public static void main(String args[])

{

   B obj = new B();

   obj.display();

}


}
```

4.Public Access modifier

The public access modifier is specified using the keyword public.

The public access modifier has the widest scope among all other access modifiers.

Classes, methods, or data members that are declared as public are accessible from everywhere in the program. There is no restriction on the scope of public data members.

```
package p1;
public class A
{
public void display()
  {
    System.out.println("GeeksforGeeks");
  }
}
package p2 ;
import p1.*;
class B {
  public static void main(String args[])
  {
    A obj = new A();
    obj.display();
} }
```

6. Write a program to copy all elements of one array into another array(Summer 23)

Ans –

```
public class CopyArray {
public static void main(String[] args) {
    //Initialize array
 int [] arr1 = new int [] {1, 2, 3, 4, 5};
  //Create another array arr2 with size of arr1
 int arr2[] = new int[arr1.length];
 //Copying all elements of one array into another
 for (int i = 0; i < arr1.length; i++) {
   arr2[i] = arr1[i];
 }
```

```
        //Displaying elements of array arr1
        System.out.println("Elements of original array: ");
        for (int i = 0; i < arr1.length; i++) {
          System.out.print(arr1[i] + " ");
        }
                        System.out.println();

        //Displaying elements of array arr2
System.out.println("Elements of new array: ");
        for (int i = 0; i < arr2.length; i++) {
          System.out.print(arr2[i] + " ");
        }
    }
}
```

Output:

```
Elements of original array
1 2 3 4 5
Elements of new array:
1 2 3 4 5
```

7. Why is circuit switching preferred over packet switching in voice communication?

Ans –

Switching is a mechanism by which data/information sent from source towards destination which are not directly connected. Networks have interconnecting devices, which receives data from directly connected sources, stores data, analyse it and then forwards to the next interconnecting device closest to the destination.

Switching can be categorized as:

- Circuit switching

Notes By – Rajan Shukla

- Packet switching
- Message switching

Circuit switching is preferred over packet switching in voice communication

because:

In circuit switching, a dedicated path is established between sender and

receiver which is maintained for entire duration of conversation.

 It provides continuous and guaranteed delivery of data.

 During the data transfer phase, no addressing is needed.

 Delays are small.

 It uses connection oriented service.

 Message received in order to the destination.

8. Describe the principles of packet switching and circuit switching techniques with neat diagram.

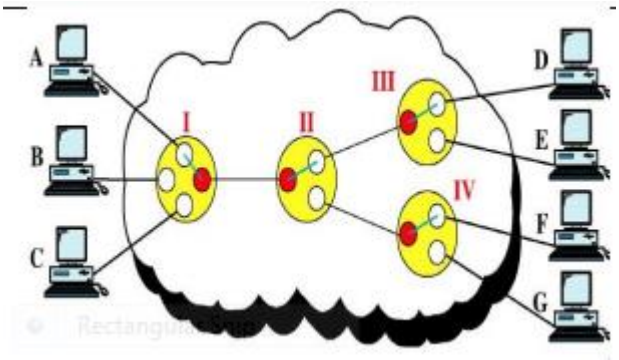
Ans –

Circuit Switching: When two nodes communicate with each other over a dedicated communication path, it is called circuit switching. There 'is a need of pre-specified route from which data will travels and no other data is permitted. In circuit switching, to transfer the data, circuit must be established so that the data transfer can take place. Circuits can be permanent or temporary. Applications which use circuit switching may have to go through three phases:

1.Establish a circuit

2.Transfer the data

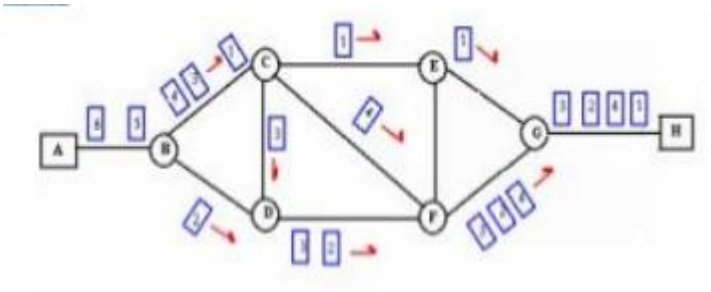
3.Disconnect the circuit



Circuit switching was designed for voice applications. Telephone is the best suitable example of circuit switching. Before a user can make a call, a virtual path between callers and called is established over the network.

Packet Switching: The entire message is broken down into smaller chunks called packets. The switching information is added in the header of each packet and transmitted independently.

It is easier for intermediate networking devices to store small size packets and they do not take much resource either on carrier path or in the internal memory of switches



Packet switching enhances line efficiency as packets from multiple applications can be multiplexed over the carrier. The internet uses packet switching technique. Packet switching enables the user to differentiate data streams based on priorities. Packets are stored and forwarded according to their priority to provide quality of service.

9. Write a program to create a class 'salary with data members empid', 'name' and 'basicsalary'. Write an interface 'Allowance' which stores rates of calculation for da as 90% of basic salary, hra as 10% of basic salary and pf as 8.33% of basic salary. Include a method to calculate net salary and display it. (Winter 22)

Ans –

```
interface allowance
{
double da=0.9*basicsalary;
double hra=0.1*basicsalary;
double pf=0.0833*basicsalary;
void netSalary();
}
class Salary
{
int empid;
String name;
float basicsalary;
Salary(int i, String n, float b)
{
empid=I;
name=n;
basicsalary =b;
}
void display()
{
System.out.println("Empid of Emplyee="+empid);
System.out.println("Name of Employee="+name);

System.out.println("Basic Salary of Employee="+ basicsalary);
}
}
class net_salary extends salary implements allowance
{
```

```
float ta;
net_salary(int i, String n, float b, float t)
{

super(i,n,b);
ta=t;
}
void disp()
{
display();
System.out.println("da of Employee="+da);
}
public void netsalary()
{
double net_sal=basicsalary+ta+hra+da;
System.out.println("netSalary of Employee="+net_sal);
}
}
class Empdetail
{
public static void main(String args[])
{
net_salary s=new net_salary(11, “abcd”, 50000);
s.disp();
s.netsalary();
}
}
```

10. Define package. How to create user defined package? Explain with example(Winter 19)(Summer 23 for 4m)

Ans –

Java provides a mechanism for partitioning the class namespace into more manageable parts. This mechanism is the package. The package is both naming and visibility controlled mechanism. Package can be created by including package as the first statement in java source code. Any classes declared within that file will belong to the specified package. Package defines a namespace in which classes are stored.

The syntax for defining a package is:

package *pkg;*

Here, pkg is the name of the package

eg : package

mypack;

Packages are mirrored by directories. Java uses file system directories to store packages. The class files of any classes which are declared in a package must be stored in a directory which has same name as package name. The directory must match with the package name exactly. A hierarchy can be created by separating package name and sub package name by a period(.) as pkg1.pkg2.pkg3; which requires a directory structure as pkg1\pkg2\pkg3.

Syntax:

To access package In a Java source file, **import** statements occur immediately following the **package** statement (if it exists) and before any class definitions .

Syntax:

import *pkg1*[.*pkg2*].(*classname*|*);

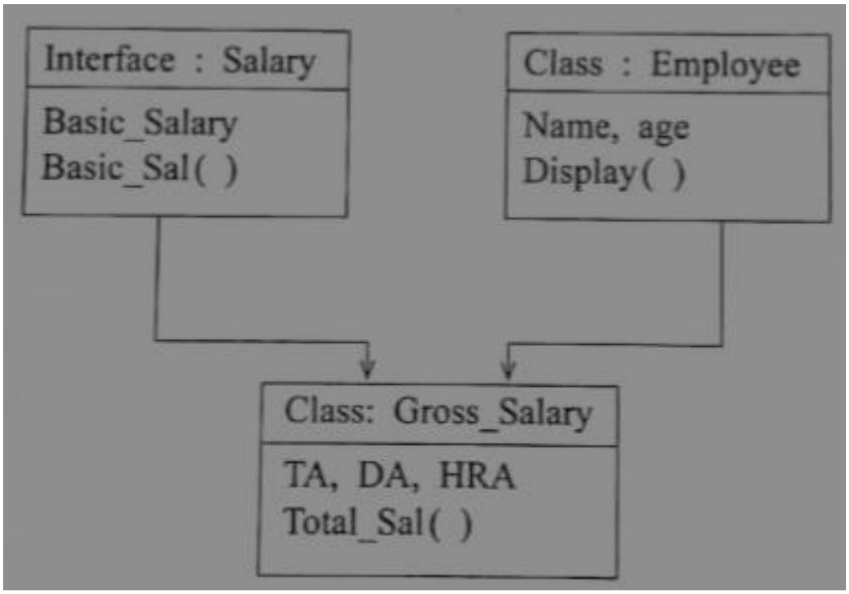
Example:

```
package package1;

public class Box

{

int l= 5;

int b = 7;

int h = 8;

public void display()

{

System.out.println("Volume is:"+(l*b*h));

}

}
```

Source file:

```
import package1.Box;

class volume

{

public static void main(String args[])

{

Box b=new Box();

b.display();

}

}
```

11. Implement the following inheritance



(Winter 19)(Summer 23 for 4 m only variables and methods are different)

Ans –

```
interface Salary
{
double Basic Salary=10000.0;
void Basic Sal();
}
class Employee
{
String Name;
int age;
Employee(String n, int b)
{
Name=n;
age=b;
}
void Display()
{
System.out.println("Name of Employee
:"+Name);
```

```
System.out.println("Age of Employee :"+age);
}
}
class Gross_Salary extends Employee implements Salary
{
double HRA,TA,DA;
Gross_Salary(String n, int b, double h,double t,double d)
{
super(n,b);
HRA=h;
TA=t;
DA=d;
}
public void Basic_Sal()
{
System.out.println("Basic Salary
:"+Basic_Salary);
}
void Total_Sal()
{
Display();
Basic_Sal();
double Total_Sal=Basic_Salary + TA + DA +
HRA;
System.out.println("Total Salary :"+Total_Sal);
}
}
class EmpDetails
{ public static void main(String args[])
{ Gross_Salary s=new
Gross_Salary("Sachin",20,1000,2000,7000);
```

```
s.Total_Sal();

}

}
```

12. Write a program to create two threads one thread will print even no. between 1 to 50 and other will print odd number between 1 to 50
Ans –

```
import java.lang.*;
class Even extends Thread
{
public void run()
{
try
{
for(int i=2;i<=50;i=i+2)
{
System.out.println("\t Even thread :"+i);
sleep(500);
}
}
catch(InterruptedException e)
{System.out.println("even thread interrupted");
}
}
}
class Odd extends Thread
{
public void run()
{
try
{
for(int i=1;i<50;i=i+2)
{
System.out.println("\t Odd thread :"+i); sleep(500);
}
}
catch(InterruptedException e)
{System.out.println("odd thread interrupted");
}
}
}
class EvenOdd
{
public static void main(String args[])
{
new Even().start();
new Odd().start();
}
}
```

13. Write a program to create two threads. One thread will display the numbers from 1 to 50 (ascending order) and other thread will display numbers from 50 to 1 (descending order).

Ans-

```
class Ascending extends Thread
{
public void run()
{
for(int i=1; i<=15;i++)
{
System.out.println("Ascending Thread : " + i);
}
}
}
class Descending extends Thread
{
public void run()
{
for(int i=15; i>0;i--) { System.out.println("Descending
Thread : " + i);
}
}
}
public class AscendingDescending Thread
{
public static void main(String[] args)
{
Ascending a=new Ascending();a.start();
Descending d=new Descending();d.start();
}
}
```

14. Write a program to input name and salary of employee and throw user defined exception if entered salary is negative.

Ans –

```
import java.io.*;
class NegativeSalaryException extends Exception
{
public NegativeSalaryException (String str)
{
super(str);
}
}
public class S1
{
public static void main(String[] args) throws IOException
{
```

```
BufferedReaderbr= new BufferedReader(new
InputStreamReader(System.in));
System.out.print("Enter Name of employee");String
name = br.readLine(); System.out.print("Enter Salary of
employee");int salary = Integer.parseInt(br.readLine());
Try
{
if(salary<0)
throw new NegativeSalaryException("Enter Salary amount
isnegative");
System.out.println("Salary is "+salary);
}
catch (NegativeSalaryException a)
{
System.out.println(a);
}
}
}
```

15. Define an exception called 'No Match Exception' that is thrown when the password accepted is not equal to "MSBTE'. Write the program. (Summer22) OR Write a program that throws an exception called "NoMatchException "whena string is not equal to "India"(Winter 23) for this refer following program

Ans –

```
import java.io.*;
class NoMatchException extends Exception
{
NoMatchException(String s)
{
super(s);
}
}
class test1
{
public static void main(String args[]) throws IOException
{
BufferedReader br= new BufferedReader(new InputStreamReader(System.in) );
System.out.println("Enter a word:");
String str= br.readLine();
try
{ if (str.compareTo("MSBTE")!=0) // can be done with equals()
throw new NoMatchException("Strings are not equal");
else
System.out.println("Strings are equal");
}
catch(NoMatchException e)
{
System.out.println(e.getMessage());
}}}
```

16. Define thread priority ? Write default priority values and the methods to set and change them.

Ans-

Thread Priority:

In java each thread is assigned a priority which affects the order in which it is scheduled for running. Threads of same priority are given equal treatment by thejava scheduler.

Default priority values as follows

The thread class defines several priority constants as: -

MIN_PRIORITY =1

NORM_PRIORITY = 5

MAX_PRIORITY = 10

Thread priorities can take value from 1-10.

getPriority(): The java.lang.Thread.getPriority() method returns the priority of thegiven thread.

setPriority(int newPriority): The java.lang.Thread.setPriority() method updates or assign the priority of the thread to newPriority. The method throws IllegalArgumentException if the value newPriority goes out of the range, which is1 (minimum) to 10 (maximum).

```
import java.lang.*;
public class ThreadPriorityExample extends Thread
{
public void run()
{
System.out.println("Inside the run() method");
}
public static void main(String argvs[])
{
ThreadPriorityExample th1 = new ThreadPriorityExample(); ThreadPriorityExample th2 =
new ThreadPriorityExample(); ThreadPriorityExample th3 = new ThreadPriorityExample();
System.out.println("Priority of the thread th1 is : " + th1.getPriority());
System.out.println("Priority of the thread th2 is : " + th2.getPriority());
System.out.println("Priority of the thread th2 is : " + th2.getPriority());th1.setPriority(6);
th2.setPriority(3);
th3.setPriority(9);
System.out.println("Priority of the thread th1 is : " + th1.getPriority());
System.out.println("Priority of the thread th2 is : " + th2.getPriority());
System.out.println("Priority of the thread th3 is : " + th3.getPriority());
System.out.println("Currently Executing The Thread : " + Thread.currentThread().gtName());
System.out.println("Priority of the main thread is : " +
Thread.currentThread().getPrority();
Thread.currentThread().setPriority(10);
System.out.println("Priority of the main thread is : " +
Thread.currentThread().getPiority());
}
```

17. Write a Java program in which thread A will display the even numbers between 1 to 50 and thread B will display the odd numbers between 1 to 50.After 3 iterations thread A should go to sleep for 500 ms.

Ans –

```
public class EvenOddThreads {
    public static void main(String[] args) {
        EvenThread evenThread = new EvenThread();
        OddThread oddThread = new OddThread();

        evenThread.start();
        oddThread.start();
    }
}

class EvenThread extends Thread {
    int count=0;
    public void run() {
        for (int i = 2; i <= 50; i += 2) {

            System.out.println("Even Thread: " + i);
if(count%3==0)
            try {
                Thread.sleep(500);

// Sleep for 500ms after every iteration
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
     count++ ;}
    }
}

class OddThread extends Thread {
    public void run() {
        for (int i = 1; i <= 50; i += 2) {
            System.out.println("Odd Thread: " + i);
            try {
                Thread.sleep(1500); // No sleep required here as even thread already
sleeps
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

18. I)Explain Errors and its types in details.
ii)Explain thread methods to set and get priority

Ans –

i) The most common errors can be broadly classified as follows:

1. Run Time Error:

Run Time errors occur or we can say, are detected during the execution of the program.

Sometimes these are discovered when the user enters an invalid data ordata which is not relevant. Runtime errors occur when a program does not contain any syntax errors but asks the computer to do something that the computer is unable to reliably do. During compilation, the compiler has no technique to detect these kinds of errors. It is the JVM (Java Virtual Machine) that detects it while the program is running. To handle the error during the run time we can put our error code inside the try block and catch the error inside the catch block.

For example: if the user inputs a data of string format when the computer isexpecting an integer, there will be a runtime error.
Example 1: Runtime Error caused by dividing by zero

2. Compile Time Error:

Compile Time Errors are those errors which prevent the code from running because of an incorrect syntax such as a missing semicolon at the end of a statement or a missing bracket, class not found, etc. These errors are detected bythe java compiler and an error message is displayed on the screen while compiling. Compile Time Errors are sometimes also referred to as **Syntax errors**

ii) There are 2 methods to set priority

1. **public final int getPriority():** java.lang.Thread.getPriority() method returnspriority of given thread.
2. **public final void setPriority(int newPriority):** java.lang.Thread.setPriority() method changes the priority ofthread to the value newPriority. This method throws IllegalArgumentException if value of parameter newPriority goes beyond minimum(1) and maximum(10) limit.

```
import java.lang.*;

// Main class
class ThreadDemo extends Thread {

// Method 1
// run() method for the thread that is called
// as soon as start() is invoked for thread in main()
public void run()
{
// Print statement System.out.println("Inside run
method");
}

// Main driver method
public static void main(String[] args)
{
// Creating random threads
// with the help of above class ThreadDemo t1 =
new ThreadDemo();ThreadDemo t2 = new
ThreadDemo();ThreadDemo t3 = new
ThreadDemo();

// Thread 1
// Display the priority of above thread
// using getPriority() method
System.out.println("t1 thread priority : "
```

```
+ t1.getPriority());

// Thread 1
// Display the priority of above thread
System.out.println("t2 thread priority : "
+ t2.getPriority());

// Thread 3
System.out.println("t3 thread priority : "
+ t3.getPriority());

// Setting priorities of above threads by
// passing integer arguments
t1.setPriority(2); t2.setPriority(5);
t3.setPriority(8);

// t3.setPriority(21); will throw
// IllegalArgumentException

// 2
System.out.println("t1 thread priority : "
+ t1.getPriority());

// 5
System.out.println("t2 thread priority : "
+ t2.getPriority());

// 8
System.out.println("t3 thread priority : "
+ t3.getPriority());

// Main thread

// Displays the name of
// currently executing Thread
System.out.println(
"Currently Executing Thread : "
+ Thread.currentThread().getName());

System.out.println( "Main thread
priority : "
+ Thread.currentThread().getPriority());

// Main thread priority is set to 10
Thread.currentThread().setPriority(10);

System.out.println( "Main thread
priority : "
+ Thread.currentThread().getPriority());
}}
```

Topic 5
Java Applets & Graphics Programming

Total Marks- 20

Specific Objectives:

The students will be able to write interactive applets and make use of graphics in programming.
They will also learn to change the background and the foreground color and to use the different fonts.

Contents:

5.1 Introduction to applets Applet, Applet life cycle (skeleton), Applet tag, Adding Applet To HTML file, passing parameter to applet, embedding <applet>tags in java code, adding controls to applets.

5.2 Graphics Programming Graphics classes, lines, rectangles, ellipse, circle, arcs, polygons, color & fonts, setColor(), getColor(), setForeGround(), setBackGround(), font class, variable defined by font class: name, pointSize, size, style, font methods: getFamily(), getFont(), getFontname(), getSize(), getStyle(), getAllFonts() & getavailablefontfamilyname() of the graphics environment class.

Topic 5 Java Applets & Graphics Programming

5.1. Introduction to applets

Applet Basics-

Basically, an applet is dynamic and interactive java program that inside the web page or applets are small java programs that are primarily used in internet computing. The java application programs run on command prompt using java interpreter whereas the java applets can be transported over the internet form one computer to another and run using the appletviewer or any web browser that supports java.

An applet is like application program which can perform arithmetic operations, display graphics, play sounds accept user input, create animation and play interactive games. To run an applet, it must be included in HTML tags for web page. Web browser is a program to view web page.

Every applet is implemented by creating sub class of Applet class. Following diagram shows the inheritance hierarchy of Applet class.

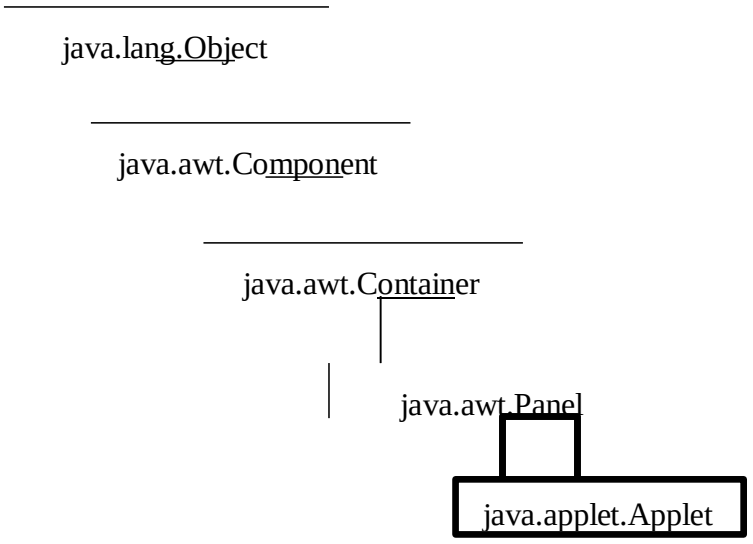
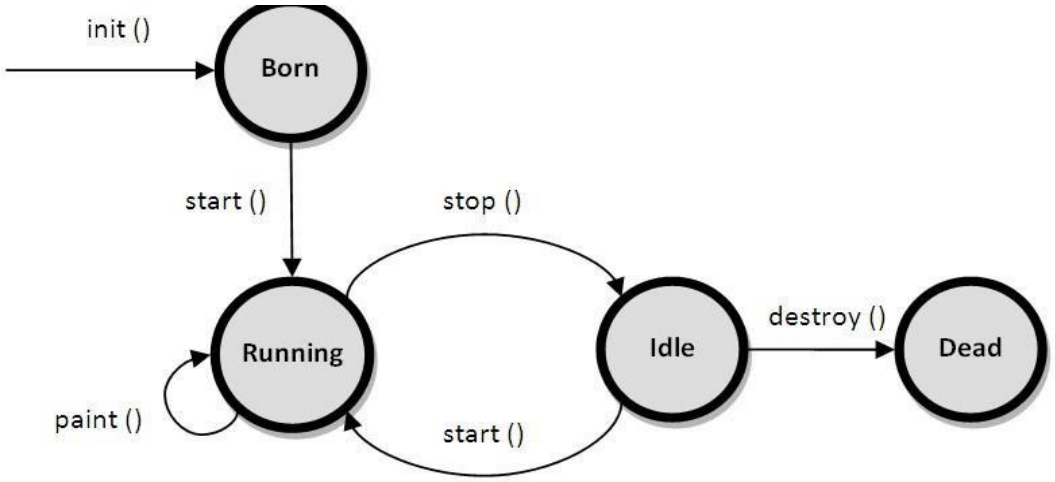


Fig. Chain of classes inherited by Applet class in java

5.1.1 Differentiate between applet and application (4 points). [W-14, S-15, W-15]

Applet	Application
Applet does not use main() method for initiating execution of code	Application use main() method for initiating execution of code
Applet cannot run independently	Application can run independently
Applet cannot read from or write to files in local computer	Application can read from or write to files in local computer
Applet cannot communicate with other servers on network	Application can communicate with other servers on network
Applet cannot run any program from local computer.	Application can run any program from local computer.
Applet are restricted from using libraries from other language such as C or C++	Application are not restricted from using libraries from other language

5.1.2 Applet life Cycle [W-14, S-15]



Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a web document.

The applet states include:

Born or initialization state
Running state
Idle state
Dead or destroyed state

a) Born or initialization state [S-15]

Applet enters the initialization state when it is first loaded. This is done by calling the init() method of Applet class. At this stage the following can be done:

Create objects needed by the applet
Set up initial values
Load images or fonts
Set up colors

Initialization happens only once in the life time of an applet.

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

b) Running state: [S-15]

Applet enters the running state when the system calls the start() method of Applet class. This occurs automatically after the applet is initialized. start() can also be called if the applet is already in idle state. start() may be called more than once. start() method may be overridden to create a thread to control the applet.

```
public void start()
        {
        //implementation
        }
```

c) Idle or stopped state:

An applet becomes idle when it is stopped from running. Stopping occurs automatically when the user leaves the page containing the currently running applet. stop() method may be overridden to terminate the thread used to run the applet.

```
public void stop()
        {
        //implementation
        }
```

d) Dead state:

An applet is dead when it is removed from memory. This occurs automatically by invoking the destroy method when we quit the browser. Destroying stage occurs only once in the lifetime of an applet. destroy() method may be overridden to clean up resources like threads.

```
public void destroy()
        {
        //implementation
        }
```

e) Display state: [S-15]

Applet is in the display state when it has to perform some output operations on the screen. This happens after the applet enters the running state. paint() method is called for this. If anything is to be displayed the paint() method is to be overridden.

```
public void paint(Graphics g)
        {
        //implementation
        }
```

5.1.3 Applet Tag & Attributes [W-15, S-16]

APPLET Tag:

The APPLET tag is used to start an applet from both an HTML document and from an applet viewer.

The syntax for the standard APPLET tag:

<APPLET

[CODEBASE = *codebaseURL*]
CODE = *appletFile*
[ALT = *alternateText*]
[NAME = *appletInstanceName*]
WIDTH = *pixels* HEIGHT = *pixels*
[ALIGN = *alignment*]
[VSPACE = *pixels*] [HSPACE = *pixels*]>
[< PARAM NAME = *AttributeName1* VALUE = *AttributeValue*>]
[<PARAM NAME = *AttributeName2* VALUE = *AttributeValue*>]
. . .

</APPLET>

CODEBASE is an optional attribute that specifies the base URL of the applet code or the directory that will be searched for the applet‟s executable class file.

CODE is a required attribute that give the name of the file containing your applet‟s compiled class file which will be run by web browser or appletviewer.

ALT: Alternate Text. The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser cannot run java applets.

NAME is an optional attribute used to specifies a name for the applet instance.

WIDTH AND HEIGHT are required attributes that give the size(in pixels) of the applet display area.

ALIGN is an optional attribute that specifies the alignment of the applet. The possible value is: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.

VSPACE AND HSPACE attributes are optional, VSPACE specifies the space, in pixels, about and below the applet. HSPACE VSPACE specifies the space, in pixels, on each side of the applet

PARAM NAME AND VALUE: The PARAM tag allows you to specifies applet-specific arguments in an HTML page applets access there attributes with the get Parameter()method.

Q. Explain <PARAM> tag of applet with suitable example. [S-15]

To pass parameters to an applet <PARAM… > tag is used. Each <PARAM…> tag has a name attribute and a value attribute. Inside the applet code, the applet can refer to that parameter by name to find its value.

The syntax of <PARAM…> tag is as follows

<PARAM NAME = name1 VALUE = value1>

To set up and handle parameters, two things must be done.

1. Include appropriate <PARAM..> tags in the HTML document.
2. Provide code in the applet to parse these parameters.

Parameters are passed on an applet when it is loaded. Generally init() method in the applet is used to get hold of the parameters defined in the <PARAM…> tag.

The getParameter() method, which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

Example

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

public class hellouser extends Applet
```

```
{
String str;
public void init()
{
str = getParameter("username");
str = "Hello "+ str;
}
public void paint(Graphics g)
{
g.drawString(str,10,100);
}
}

<HTML>
<Applet code = —hellouser.class‖ width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</Applet>
</HTML>
```

Q. How can parameter be passed to an applet? Write an applet to accept user name in the form of parameter and print „Hello<username>". [W-15]

Passing Parameters to Applet

User defined parameters can be supplied to an applet using <PARAM…..> tags. PARAM tag names a parameter the Java applet needs to run, and provides a value for that parameter.

PARAM tag can be used to allow the page designer to specify different colors, fonts, URLs or other data to be used by the applet.

To set up and handle parameters, two things must be done.

1. Include appropriate <PARAM..>tags in the HTML document.

The Applet tag in HTML document allows passing the arguments using param tag.

The syntax of <PARAM…> tag

<Applet code=”AppletDemo” height=300 width=300>
<PARAM NAME = name1 VALUE = value1>
</Applet>

NAME: attribute name
VALUE: value of attribute named by corresponding PARAM NAME.

2. Provide code in the applet to parse these parameters.

The Applet access their attributes using the getParameter method.
The syntax is : **String getParameter(String name);**

Program for an applet to accept user name in the form of parameter and print „Hello<username>‟ [W-15]

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

public class hellouser extends Applet
{
String str;
public void init()
{
str = getParameter("username");
str = "Hello "+ str;
}
public void paint(Graphics g)
{
g.drawString(str,10,100);
}
}
```

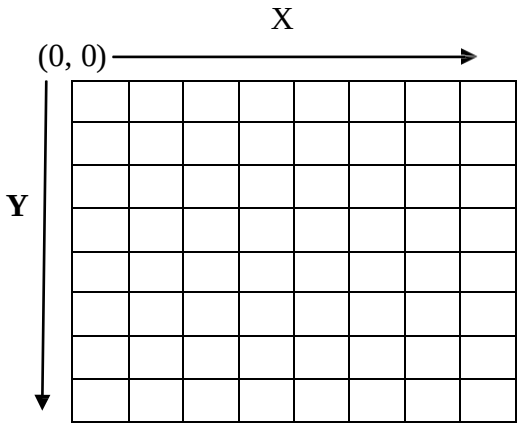
```
<HTML>
<Applet code = —hellouser.class‖ width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</Applet>
</HTML>
```

5.2. Graphics Programming

Graphics can be drawn with the help of java. java applets are written to draw lines, figures of different shapes, images and text in different styles even with the colours in display.

Every applet has its own area of the screen known as canvas, where it creates the display in the area specified the size of applet‘s space is decided by the attributes of <APPLET...> tag.

A java applet draws graphical image inside its space using the coordinate system shown in following fig., which shows java‘s coordinate system has the origin (0, 0) in the upper-left corner, positive x values are to be right, and positive y values are to the bottom. The values of coordinates x and y are in pixels.



Q. Write a simple applet which display message „Welcome to Java". [W-15]

Program:

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

public class Welcome extends Applet
   {
   public void paint( Graphics g)
            {
             g.drawString(—Welcome to java‖,25,50);
            }
   }

/*<applet code= WelcomeJava width= 300 height=300>
</applet>*/
```

Step to run an Applet

1. Write a java applet code and save it with as a class name declared in a program by extension as a .java.
   e.g. from above java code file we can save as a **Welcome.java**

2. Compile the java file in command prompt jdk as shown below
   C:\java\jdk1.7.0\bin> javac Welcome.java

3. After successfully compiling java file, it will create the .class file, e.g Welcome.class. then we have to write applet code to add this class into applet.

4. Applet code

   ```
   <html>
   <Applet code= — Welcome.class‖ width= 500 height=500>
   </applet>
   </html>
   ```

5. Save this file with Welcome.html in _bin' library folder.

6. Now write the following steps in command prompt jdk.

   ```
   C:\java\jdk1.7.0\bin>  appletviewer  Welcome.java
   C:\java\jdk1.7.0\bin>  appletviewer  Welcome.html
   (Shows output in applet viewer)
   OR
   C:\java\jdk1.7.0\bin> Welcome.html
   (Shows output in internet browser)
   ```

5.2.1. Graphics Class

The Graphics class of java includes methods for drawing different types of shapes, from simple lines to polygons to text in a variety of fonts.

The paint( ) method and a Graphics object is used to display text. To draw shapes, drawing methods in Graphics class is used which arguments representing end points, corners, or starting locations of a shape as a values in the applet's coordinate system.

Method	Description
clearRect( )	Erases a rectangular area of the canvas
copyArea( )	Copies a rectangular area of the canvas to another area
drawArc( )	Draws a hollow arc.
drawLine( )	Draws a straight line
drawOval( )	Draws a hollow oval
drawPolygon( )	Draws a hollow polygon
drawRect( )	Draws a hollow rectangle
drawRoundRect( )	Draws a hollow rectangle with rounded corners.
drawstring( )	Displays a text string
fillArc( )	Draws a filled arc
fillOval( )	Draws a filled arc
fillPolygon( )	Draws a filled polygon
fillRect( )	Draws a filled rectangle
fillRoundRect( )	Draws filled rectangle with rounded corners
getColor( )	Retrieves the current drawing color
getFont( )	Retrieves the currently used font
getFontMetrics( )	Retrieves information about the current font.
setColor( )	Sets the drawing color
setFont( )	Seta fonts.

5.2.2. drawString( ) [S-15]

Displaying String:
drawString() method is used to display the string in an applet window

Syntax:

void drawString(String message, int x, int y);
where message is the string to be displayed beginning at x, y

Example:

g.drawString(—WELCOME‖, 10, 10);

5.2.3. Lines and Rectangle.

5.2.3.1. drawLine( )

The drawLine ( ) method is used to draw line which takes two pair of coordinates (x1,y1) and (x2, y2) as arguments and draws a line between them. The graphics object g is passed to paint( ) method.

The syntax is

```
g.drawLine(x1,y1,x2,y2);
```

e.g. g.drawLine(20,20,80,80);

5.2.3.2. drawRect( ) [W-14, S-15,W-15, S-16]

The drawRect() method display an outlined rectangle

Syntax: void drawRect(int top, int left, int width, int height)

This method takes four arguments, the first two represents the x and y co- ordinates of the top left corner of the rectangle and the remaining two represent the width and height of rectangle.

Example: g.drawRect(10,10,60,50);

Q. Design an Applet program which displays a rectangle filled with red color and message as "Hello Third year Students" in blue color. [S-16]

Program-

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

public class DrawRectangle extends Applet
        {
        public void paint(Graphics g)
                {
                        g.setColor(Color.red);
                        g.fillRect(10,60,40,30);

                        g.setColor(Color.blue);
                        g.drawString("Hello Third year Students",70,100);
                }
        }
```

```
/* <applet code="DrawRectangle.class" width="350" height="300"> </applet> */
```

5.2.4. Circle and Ellipse

5.2.4.1. drawOval( ) [W-14, S-15, W-15, S-16]

To draw an Ellipses or circles used drawOval() method can be used.

Syntax: void drawOval(int top, int left, int width, int height)

The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by top and left and whose width and height are specified by width and height to draw a circle or filled circle, specify the same width and height the following program draws several ellipses and circle.

Example: g.drawOval(10,10,50,50);

5.2.4.2. fillOval ( ) [W-14]

Draws an oval within a bounding rectangle whose upper left corner is specified by top, left. Width and height of the oval are specified by width and height.

Syntax- void fillOval(int top, int left, int width, int height):

Example g.fillOval(10,10,50,50);

Q. Write a simple applet program which display three concentric circle. [S-16]

Program-

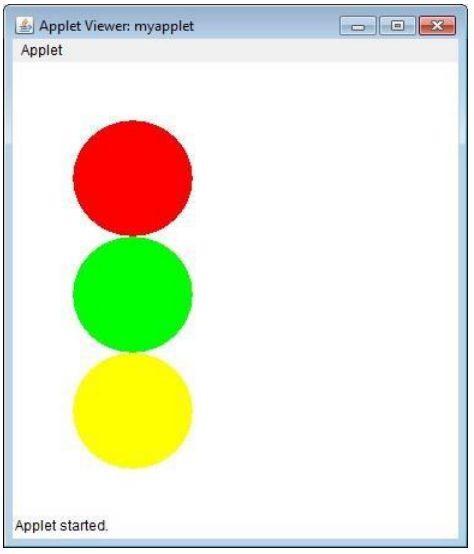
```
import java.awt.*;
import java.applet.*;

public class CircleDemo extends Applet
        {
            public void paint (Graphics g)
                    {

                    g.drawOval(100,100,190,190);
                    g.drawOval(115,115,160,160);
                    g.drawOval(130,130,130,130);
                    }
            }


/*<applet code="CircleDemo.class" height=300 width=200>
 </applet>*/

(OR)

HTML Source:
<html> <applet code="CircleDemo.class" height=300 width=200>
</applet>
</html>
```

Q. Write a program to design an applet to display three circles filled with three different colors on screen. [W-14, W-15]

Program-

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

public class myapplet extends Applet
        {
                public void paint(Graphics g)
                { g.setColor(Color.red);
                g.fillOval(50,50,100,100);

                g.setColor(Color.green);
                g.fillOval(50,150,100,100);

                g.setColor(Color.yellow);
                g.fillOval(50,250,100,100);

                }
        }

/*<applet code=myapplet width= 300 height=300>
</applet>*/
```

Output



5.2.5. Drawing Arcs

5.2.5.1. drawArc( ) [S-15, W-15]

It is used to draw arc

Syntax:

```
void drawArc(int x, int y, int w, int h, int  start_angle, int  sweep_angle);
```

where x, y starting point, w& h are width and height of arc, and start_angle is starting angle of arc sweep_angle is degree around the arc

Example:

```
g.drawArc(10, 10, 30, 40, 40, 90);
```

5.2.6. Drawing polygons

5.2.6.1. drawPolygon( ) [W-14, W-15]

drawPolygon() method is used to draw arbitrarily shaped figures.

Syntax- void drawPolygon(int[ ] xPoints, int[ ] yPoints, int numPoints):

The polygon‟s end points are specified by the co-ordinates pairs contained within the x and y arrays. The number of points define by x and y is specified by numPoints.

Example-

```
int  x[ ] = {10, 170, 80};
int  y[ ] = {20, 40, 140};
int  n = 3;

g.drawPolygon(x, y, n);
```

Q. Write the syntax and example for each of following graphics methods:

1) drawPoly ( ) 2) drawRect ( ) 3) drawOval ( ) 4) fillOval ( )

For syntax refer above 5.2.3.2 and all.....

Example for including all methods in a one program

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

public class DrawGraphics extends Applet
{
        public void paint(Graphics g)
        {
        int x[ ] = {10, 170, 80};
        int y[ ] = {20, 40, 140};
        int n = 3;
        g.drawPolygon(x, y, n);
        g.drawRect(10, 150,100, 80);
        g.drawOval(10, 250, 100, 80);
        g.fillOval(10, 350, 100, 80);
        }
}

/*
<applet code = DrawGraphics.class height = 500 width = 400>
</applet>*/
```

5.2.7. Setting color of an Applet

Background and foreground color of an applet can be set by using followings methods

void setBackground(Color.newColor)

void setForeground (Color.newColor)

where newColor specifies the new color. The class color defines the constant for specific color listed below.

Color.black	Color.white	Color.pink	Color.yellow
Color.lightGray	Color.gray	Color.darkGray	Color.red
Color.green	Color.magenda	Color.orange	Color.cyan

Example

setBackground(Color.red);

setForeground (Color.yellow);

The following methods are used to retrieve the current background and foreground color.

Color getBackground( )
Color getForeground( )

5.2.8. Font class

A font determines look of the text when it is painted. Font is used while painting text on a graphics context & is a property of AWT component.

The Font class defines these variables:

Variable	Meaning
String name	Name of the font
float pointSize	Size of the font in points
int size	Size of the font in point
int style	Font style

5.2.8.1. Use of font class [W-14, S-15]

The Font class states fonts, which are used to render text in a visible way. It is used to set or retrieve the screen font.

Syntax to create an object of Font class. [W-14]

To select a new font, you must first construct a Font object that describes that font. Font constructor has this general form:

Font(String fontName, int fontStyle, int pointSize)

fontName specifies the name of the desired font. The name can be specified using either the logical or face name.

All Java environments will support the following fonts:

Dialog, DialogInput, Sans Serif, Serif, Monospaced, and Symbol. Dialog is the font used by once system‟s dialog boxes.

Dialog is also the default if you don‟t explicitly set a font. You can also use any other fonts supported by particular environment, but be careful—these other fonts may not be universally available.

The style of the font is specified by fontStyle. It may consist of one or more of these three constants:

Font.PLAIN, Font.BOLD, and Font.ITALIC. To combine styles, OR them together.

For example,

Font.BOLD | Font.ITALIC specifies a bold, italics style.

The size, in points, of the font is specified by pointSize.

To use a font that you have created, you must select it using setFont( ), which is defined by Component.

It has this general form:

void setFont(Font fontObj)

Here, fontObj is the object that contains the desired font

5.2.8.2. Methods of font class

Q. Describe any three methods of font class with their syntax and example of each. [W-14, S-15]

Sr. No	Methods	Description
1	static Font decode(String *str*)	Returns a font given its name.
2	boolean equals(Object *FontObj*) :	Returns **true** if the invoking object contains the same font as that specified by *FontObj*.Otherwise, it returns **false**.
3	String toString( )	Returns the string equivalent of the invoking font.
4	String getFamily( )	Returns the name of the font family to which the invoking font belongs.
5	static Font getFont(String *property*)	Returns the font associated with the system property specified by *property*. **null** is returned if *property* does not exist.
6	static Font getFont(String *property*,Font *defaultFont*)	Returns the font associated with the System property specified by *property*. The font specified by *defaultFont* is returned if *property* does not exist.
7	String getFontName( )	Returns the face name of the invoking font.
8	String getName( )	Returns the logical name of the invoking font.

9	int getSize( )	Returns the size, in points, of the invoking font.
10	int getStyle( )	Returns the style values of the invoking font.
11	int hashCode( )	Returns the hash code associated with the invoking object.
12	boolean isBold( )	Returns **true** if the font includes the **BOLD** style value. Otherwise, **false** is returned.
13	boolean isItalic( )	Returns **true** if the font includes the **ITALIC** style value. Otherwise, **false** is returned.
14	boolean isPlain( )	Returns **true** if the font includes the **PLAIN** style value. Otherwise, **false** is returned.

Example:-

//program using equals method

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

public class ss extends Applet
        {
        public void paint(Graphics g)
                {
                Font a = new Font ("TimesRoman", Font.PLAIN, 10);
                Font b = new Font ("TimesRoman", Font.PLAIN, 10);

                // displays true since the objects have equivalent settings
                g.drawString(""+a.equals(b),30,60);
                }
        }

/*<applet code=‖ss.class‖ height=200 width=200>
</applet>*/
```

// program using getFontName,getFamily(),getSize(),getStyle(),.getName()

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

public class font1 extends Applet
        {
        Font f, f1;
        String s, msg;
        String  fname;
        String  ffamily;
        int size;
        int style;
        public void init()
                {
```

```
                f= new Font("times new roman",Font.ITALIC,20);
                setFont(f);
                msg="is interesting";
                s="java programming";
                fname=f.getFontName();
                ffamily=f.getFamily();
                size=f.getSize();
                style=f.getStyle();
                String f1=f.getName();
                }
        public void paint(Graphics g)
                {
                g.drawString("font name"+fname,60,44);
                g.drawString("font family"+ffamily,60,77);
                g.drawString("font size "+size,60,99);
                g.drawString("fontstyle "+style,60,150);
                g.drawString("fontname "+f1,60,190);
                }
        }

/*<applet code=font1.class height=300 width=300>
</applet>*/
```

Q. Write method to set font of a text and describe its parameters. [S-16]

The AWT supports multiple type fonts emerged from the domain of traditional type setting to become an important part of computer-generated documents and displays. The AWT provides flexibility by abstracting font-manipulation operations and allowing for dynamic selection of fonts.

Fonts have a family name, a logical font name, and a face name. The family name is the general name of the font, such as Courier. The logical name specifies a category of font, such as Monospaced. The face name specifies a specific font, such as Courier Italic To select a new font, you must first construct a Font object that describes that font.

One Font constructor has this general form:

Font(String fontName, intfontStyle, intpointSize)

To use a font that you have created, you must select it using setFont( ), which is defined by Component.

It has this general form:

void setFont(Font fontObj)

Example

```
        import java.applet.*;
        import  java.awt.*;
        import  java.awt.event.*;
        public class SampleFonts extends Applet
                {
                int next = 0;
                Font  f;
                String msg;
```

```
public void init( )
        {
        f = new Font("Dialog", Font.PLAIN, 12);
        msg = "Dialog";
        setFont(f);
public void paint(Graphics g)
        {
        g.drawString(msg, 4, 20);
        }
}
```

Q. State purpose of get Available Font Family Name ( ) method of graphics environment class.

Purpose of getAvailableFontFamilyName() method:

It returns an array of String containing the names of all font families in this Graphics Environment localized for the specified locale

Syntax:
public abstract String[ ] getAvailableFontFamilyNames(Locale 1)

Parameters:

l - a Locale object that represents a particular geographical, political, or cultural region. Specifying null is equivalent to specifying Locale.getDefault().

Or

String[ ] getAvailableFontFamilyNames( )

It will return an array of strings that contains the names of the available font families

Important Questions:-

4 Marks Questions:-

1) Write syntax and example of 1) drawString ( ) 2) drawRect ( ) ; 3) drawOval ( ) 4) drawArc ( ).
2) Describe following states of applet life cycle : a) Initialization state. b) Running state. c) Display state
3) State the use of font class. Describe any three methods of font class with their syntax and example of each.
4) Differentiate applet and application with any four points.
5) State syntax and explain it with parameters for : i)drawRect ( ) ii) drawOral ( )
6) Design an Applet program which displays a rectangle filled with red color and message as ―Hello Third year Students‖ in blue color.
7) Describe applet life cycle with suitable diagram.
8) Differentiate between applet and application (any 4 points).
9) Write a program to design an applet to display three circles filled with three different colors on screen.
10) Explain all attributes available in < applet > tag.

6 & 8 Marks Questions:-

1) Explain <PARAM> Tag of applet with suitable example.
2) State the use of font class. Describe any three methods of font class with their syntax and example of each.
3) Write a simple applet program which display three concentric circle.
4) Write method to set font of a text and describe its parameters.
5) Explain <applet> tag with its major attributes only. State purpose of get Available Font Family Name ( ) method of graphics environment class.
6) Design an applet which displays three circles one below the other and fill them red, green and yellow color respectively.
7) Write the syntax and example for each of following graphics methods : 1) drawPoly ( ) 2) drawRect ( ) 3) drawOval ( ) 4) fillOval ( )
8) State the use of Font class. Write syntax to create an object of Font class.
9) Describe any 3 methods of Font class with their syntax and example of each.
10) Write syntax and example of following Graphics class methods : (i) drawOval( ) (ii) drawPolygon( ) (iii) drawArc( ) (iv) drawRect( )
11) Differentiate between applet and application and also write a simple applet which display message _Welcome to Java‘.
12) How can parameters be passed to an applet ? Write an applet to accept user name in the form of parameter and print _Hello < username >‘.

Topic 6- File I/O & collection frame work

Total Marks- 12

6.1. Stream Classes

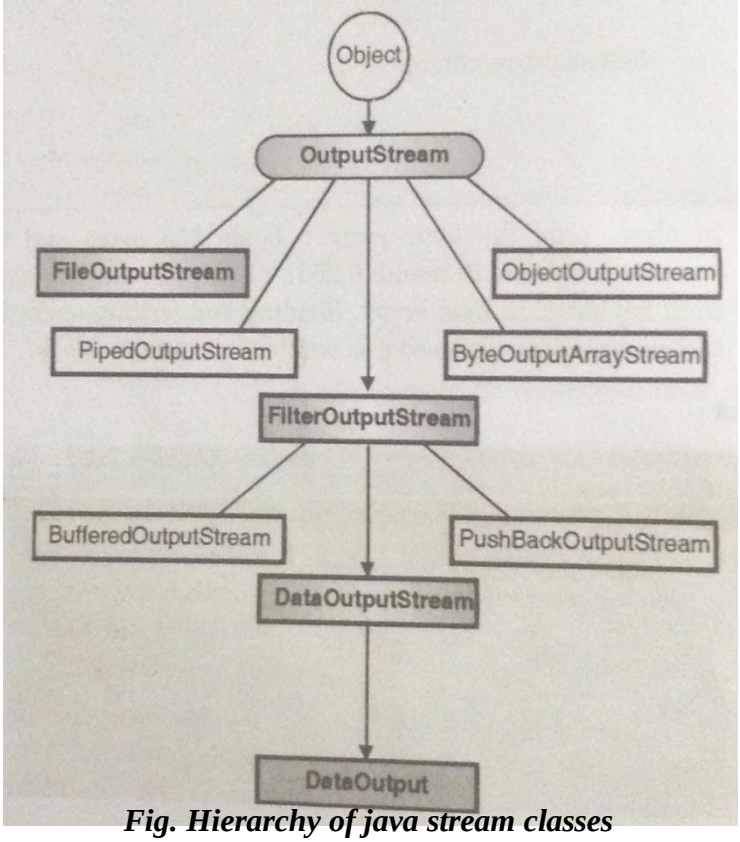


Fig. Hierarchy of java stream classes

1. **What are stream classes ? List any two input stream classes from character stream [S-15, S-16]**

Definition:

The java. IO package contain a large number of stream classes that provide capabilities for processing all types of data. These classes may be categorized into two groups based on the data type on which they operate.

1. Byte stream classes that provide support for handling I/O operations on bytes.
2. Character stream classes that provide support for managing I/O operations on characters.

Character Stream Class can be used to read and write 16-bit Unicode characters. There are two kinds of character stream classes, namely, reader stream classes and writer stream classes

Reader stream classes:-

It is used to read characters from files. These classes are functionally similar to the input stream classes, except input streams use bytes as their fundamental unit of information while reader streams use characters

Input Stream Classes

1. BufferedReader
2. CharArrayReader
3. InputStreamReader
4. FileReader
5. PushbackReader
6. FilterReader
7. PipeReader
8. StringReader

2. What are streams ? Write any two methods of character stream classes. [W-15]

Java programs perform I/O through streams. A stream is an abstraction that either produces or consumes information (i.e it takes the input or gives the output). A stream is linked to a physical device by the Java I/O system.

All streams behave in the same manner, even if the actual physical devices to which they are linked differ. Thus, the same I/O classes and methods can be applied to any type of device.

Java 2 defines two types of streams: byte and character.

Byte streams provide a convenient means for handling input and output of bytes. Byte streams are used, for example, when reading or writing binary data.

Character streams provide a convenient means for handling input and output of characters.

They use Unicode and, therefore, can be internationalized. Also, in some cases, character streams are more efficient than byte streams.

The Character Stream Classes

Character streams are defined by using two class hierarchies. At the top are two abstract classes, **Reader** and **Writer**. These abstract classes handle Unicode character streams. Java has several concrete subclasses of each of these.

Methods of Reader Class

1) void mark(int *numChars*) : Places a mark at the current point in the input stream that will remain valid until numChars characters are read.

2) boolean markSupported( ) : Returns **true** if **mark( )** / **reset( )** are supported on this stream.

3) int read( ) : Returns an integer representation of the next available character from the invoking input stream. –1 is returned when the end of the file is encountered.

4) int read(char *buffer*[ ]) : Attempts to read up to buffer. Length characters into buffer and returns the actual number of characters that were successfully read. –1 is returned when the end of the file is encountered.

5) abstract int read(char *buffer*[ ],int *offset*,int *numChars*): Attempts to read up to numChars characters into buffer starting at buffer[offset], returning the number of characters successfully read.–1 is returned when the end of the file is encountered.

6) boolean ready( ): Returns **true** if the next input request will not wait. Otherwise, it returns **false**.

7) void reset( ): Resets the input pointer to the previously set mark.

8) long skip(long *numChars*) :- Skips over *numChars* characters of input, returning the number of characters actually skipped.

9) abstract void close( ) :- Closes the input source. Further read attempts will generate an IOException

Writer Class

Writer is an abstract class that defines streaming character output. All of the methods in this class return a **void** value and throw an **IOException** in the case of error

Methods of Writer class are listed below: -

1) abstract void close( ) : Closes the output stream. Further write attempts will generate an **IOException**.

2) abstract void flush( ) : Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.

3) void write(int*ch*): Writes a single character to the invoking output stream. Note that the parameter is an **int**, which allows you to call **write** with expressions without having to cast them back to **char**.

4) void write(char *buffer*[ ]): Writes a complete array of characters to the invoking output stream

5) abstract void write(char *buffer*[ ],int *offset*, int *numChars*) :- Writes a subrange of *numChars* characters from the array *buffer*, beginning at *buffer*[*offset*] to the invoking output stream.

6) void write(String *str*): Writes *str* to the invoking output stream.

7) void write(String *str*, int *offset*,int *numChars*): Writes a sub range of *numChars* characters from the array *str*, beginning at the specified *offset*.

[Note: any two methods from above list to be considered]**

3. What is use of stream classes ? Write any two methods FileReader class.[W-14]

An I/O Stream represents an input source or an output destination.

A stream can represent many different kinds of sources and destinations, including disk files, devices, other programs, and memory arrays.

Streams support many different kinds of data, including simple bytes, primitive data types, localized characters, and objects.

Some streams simply pass on data; others manipulate and transform the data in useful ways. Java‟s stream based I/O is built upon four abstract classes: InputStream, OutputStream, Reader, Writer.

They are used to create several concrete stream subclasses, the top level classes define the basic functionality common to all stream classes.

InputStream and OutputStream are designed for byte streams and used to work with bytes or other binary objects.

Reader and Writer are designed for character streams and used to work with character or string.

1. public int read()throws IOException - Reads a single character.

2. public int read(char[ ] cbuf, int offset, int length) throws IOException - Reads characters into a portion of an array.

3. public void close()throws IOException - Closes the stream and releases any system resources associated with it. Once the stream has been closed, further read(), ready(), mark(), reset(), or skip() invocations will throw an IOException. Closing a previously closed stream has no effect

4. public boolean ready()throws IOException - Tells whether this stream is ready to be read. An InputStreamReader is ready if its input buffer is not empty, or if bytes are available to be read from the underlying byte stream

5. public void mark(int readAheadLimit) throws IOException -Marks the present position in the stream. Subsequent calls to reset() will attempt to reposition the stream to this point. Not all character-input streams support the mark() operation.

6. public void reset()throws IOException - Resets the stream. If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the reset() operation, and some support reset() without supporting mark().

4. Write any two methods of File and FileInputStream class each.[S-15, W-15]

File Class Methods

1. **String getName( )** - returns the name of the file.

2. **String getParent( )** - returns the name of the parent directory.

3. boolean exists( ) - returns true if the file exists, false if it does not.

4. void deleteOnExit( ) -Removes the file associated with the invoking object when the Java Virtual Machine terminates.

5. boolean isHidden( )-Returns true if the invoking file is hidden. Returns false otherwise.

FileInputStream Class Methods:

1. int available( )- Returns the number of bytes of input currently available for reading.

2. void close( )- Closes the input source. Further read attempts will generate an IOException.

3. void mark(int numBytes) -Places a mark at the current point in the inputstream that will remain valid until numBytes bytes are read.

4. boolean markSupported( ) -Returns true if mark( )/reset( ) are supported by the invoking stream.

5. int read( )- Returns an integer representation of the next available byte of input. –1 is returned when the end of the file is encountered.

6. int read(byte buffer[ ])- Attempts to read up to buffer.length bytes into buffer and returns the actual number of bytes that were successfully read. –1 is returned when the end of the file is encountered.

5. Explain serialization in relation with stream class. [W-14,W-15, S-16]

Serialization is the process of writing the state of an object to a byte stream. This is useful when you want to save the state of your program to a persistent storage area, such as a file. At a later time, you may restore these objects by using the process of deserialization.

Serialization is also needed to implement Remote Method Invocation (RMI). RMI allows a Java object on one machine to invoke a method of a Java object on a different machine. An object may be supplied as an argument to that remote method. The sending machine serializes the object and transmits it. The receiving machine deserializes it.

Example:

Assume that an object to be serialized has references to other objects, which, inturn, have references to still more objects. This set of objects and the relationships among them form a directed graph. There may also be circular references within this object graph. That is, object X may contain a reference to object Y, and object Y may contain a reference back to object X. Objects may also contain references to themselves. The object serialization and deserialization facilities have been designed to work correctly in these scenarios. If you attempt to serialize an object at the top of an object graph, all of the other referenced objects are recursively located and serialized. Similarly, during the process of deserialization, all of these objects and their references are correctly restored.

6. Write a program to copy contents of one file to another file using character stream class.[S-15]

```
import java.io.*;
class CopyData
{
        public static void main(String args[ ])
        {

        //Declare input and output file stream

        FileInputStream  fis= null;  //input stream

        FileOutputStream  fos=null;  //output Stream
        //Declare a variable to hold a byte

        byte  byteRead;
         try
        {
        // connect fis to in.dat
        fis=new FileInputSream(—in.dat‖);
        // connect fos to out.dat
        fos= new FileOutputStream(—out.dat‖);
        //reading bytes from in.dat and write to out.dat

        do
        {
        byteRead =(byte)fis.read( );
        fos.write(byteRead);
        }
        while(byteRead != -1);
        }

        Catch(FileNotFoundException e)

        {

        System.out.println(—file not found‖);

        }

        Catch(IOException  e)

        {

        System.out.pritln(e.getMessage( ));

        }

        finally     // close file

        {
```

```
        try
        {
        fis.close( );
        fos.close( );
        }
        Catch(IOException  e)
        {  }
        }
        }
}
```

7. **What is use of ArrayList Class ? State any three methods with their use from ArrayList.[W-15, S-16]**

Use of ArrayList class:

1. ArrayListsupports dynamic arrays that can grow as needed.
2. ArrayListis a variable-length array of object references. That is, an ArrayListcan dynamically increase or decrease in size. Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Methods of ArrayList class :

1. void add(int index, Object element) Inserts the specified element at the specified position index in this list. Throws IndexOutOfBoundsException if the specified index is is out of range (index < 0 || index >size()).

2. boolean add(Object o) Appends the specified element to the end of this list.

3. booleanaddAll(Collection c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator. Throws NullPointerException if the specified collection is null.

4. booleanaddAll(int index, Collection c) Inserts all of the elements in the specified collection into this list, starting at the specified position. Throws NullPointerException if the specified collection is null.

5. void clear() Removes all of the elements from this list.

6. Object clone() Returns a shallow copy of this ArrayList.

7. boolean contains(Object o) Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element e such that (o==null ? e==null : o.equals(e)).

8. void ensureCapacity(intminCapacity) Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

9. Object get(int index) Returns the element at the specified position in this list. Throws IndexOutOfBoundsException if the specified index is is out of range (index <
0 || index >= size()).

10. intindexOf(Object o) Returns the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.

11. intlastIndexOf(Object o) Returns the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.

12. Object remove(int index) Removes the element at the specified position in this list. Throws IndexOutOfBoundsException if index out of range (index < 0 || index >= size()).

13. protected void removeRange(intfromIndex, inttoIndex) Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.

14. Object set(int index, Object element) Replaces the element at the specified position in this list with the specified element. Throws IndexOutOfBoundsException if the specified index is is out of range (index < 0 || index >= size()).

15. int size() Returns the number of elements in this list.

16. Object[ ] toArray() Returns an array containing all of the elements in this list in the correct order. Throws NullPointerException if the specified array is null.

17. Object [] toArray(Object[] a) Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

18. void trimToSize() Trims the capacity of this ArrayList instance to be the list's current size.

8. Write syntax and function of following methods of Date class :
i) getTime ( ) ii) getDate ( ) [S-15]

The **Date** class encapsulates the current date and time.

i. getTime():

Syntax:

long getTime( )

Returns the number of milliseconds that have elapsed since January 1, 1970.

ii. getDate()

Syntax:

public int getDate()

Returns the day of the month. This method assigns days with the values of 1 to 31.

9. Write syntax and function of following methods of date class :
1) sethme ( ) 2) getDay ( ) [W-14]

1. setTime():

void setTime(long time):
the parameter time - the number of milliseconds.
Sets this Date object to represent a point in time that is time milliseconds after January 1, 1970 00:00:00 GMT

2. getDay()

int getDay():
Returns the day of the week represented by this date.
The returned value (0 = Sunday, 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday) represents the day of the week that contains or begins with the instant in time represented by this Date object, as interpreted in the local time zone.

10. Write any four mathematical functions used in Java.[W-14]

1) min() :
Syntax: static int min(int a, int b)
Use: This method returns the smaller of two int values.

2) max() :
Syntax: static int max(int a, int b)
Use: This method returns the greater of two int values.

3) sqrt()
Syntax: static double sqrt(double a)
Use : This method returns the correctly rounded positive square root of a double

value.

4) **pow()** :
Syntax: static double pow(double a, double b)
Use : This method returns the value of the first argument raised to the power of the second argument.

5) exp()
Syntax: static double exp(double a)
Use : This method returns Euler's number e raised to the power of a double value.

6) round() :
Syntax: static int round(float a)

Use : This method returns the closest int to the argument.

7) abs()
Syntax: static int abs(int a)
Use : This method returns the absolute value of an int value.

11. What is use of setclass ? Write a program using setclass.[W-14]

The Set interface defines a set. It extends Collection and declares the behavior of a collection that does not allow duplicate elements. Therefore, the add( ) method returns false if an attempt is made to add duplicate elements to a set.

The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

The methods declared by Set are summarized in the following table

Sr.No	Methods	Description
1	add()	Adds an object to the collection
2	clear()	Removes all objects from the collection
3	contains()	Returns true if a specified object is an element within the collection
4	isEmpty()	Returns true if the collection has no elements
5	iterator()	Returns an Iterator object for the collection which may be used to retrieve an object
6	remove()	Removes a specified object from the collection
7	size()	Returns the number of elements in the collection

Following is the example to explain Set functionality:

```
import java.util.*;
public class SetDemo
        {
        public static void main(String args[])
        {
        int count[] = {34, 22,10,60,30,22};
        Set<Integer> set = new HashSet<Integer>();
        try{
        for(int i = 0; i<5; i++){
        set.add(count[i]);
        }
        System.out.println(set);
        TreeSet sortedSet = new TreeSet<Integer>(set);
        System.out.println("The sorted list is:");
        System.out.println(sortedSet);
System.out.println("The First element of the set is: "+ (Integer)sortedSet.first());

System.out.println("The last element of the set is: "+ (Integer)sortedSet.last());
}
catch(Exception e){}
}
}
```

Executing the program.

```
[34, 22, 10, 30, 60]
The sorted list is:
[10, 22, 30, 34, 60]
The First element of the set is: 10
The last element of the set is: 60
```

12. State syntax and describe any two methods of map class.[S-16]

The Map Classes Several classes provide implementations of the map interfaces. A map is an object that stores associations between keys and values, or key/value pairs. Given a key, you can find its value. Both keys and values are objects. The keys must be unique, but the values may be duplicated. Some maps can accept a null key and null values, others cannot.

Methods:

void clear // removes all of the mapping from map

booleancontainsKey(Object key) //Returns true if this map contains a mapping for the specified key.

Boolean conainsValue(Object value)// Returns true if this map maps one or more keys to the specified value

Boolean equals(Object o) //Compares the specified object with this map for equality