WINTER – 2022 EXAMINATION

Subject Name: Java Programming **Model Answer** **Subject Code:** 22412

Important Instructions to examiners:

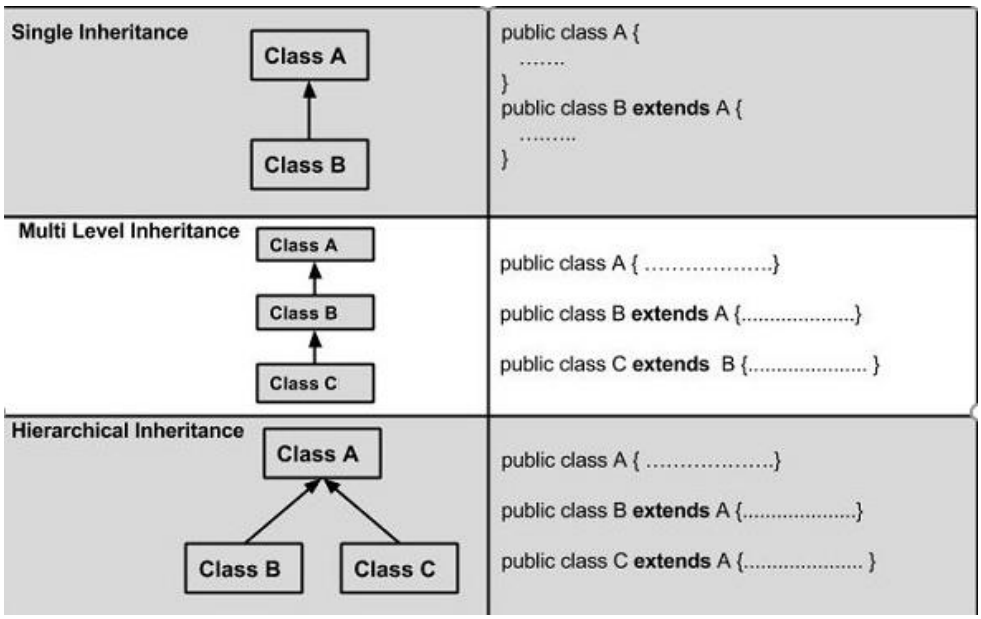
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

<table>
<tr><th>Q. No.</th><th>Sub Q. N.</th><th>Answer</th><th>Marking Scheme</th></tr>
<tr><td>1</td><td></td><td>Attempt any FIVE of the following:</td><td>10 M</td></tr>
<tr><td></td><td>a)</td><td>State any four relational operators and their use.</td><td>2 M</td></tr>
<tr><td></td><td>Ans</td><td><table><tr><th>Operator</th><th>Meaning</th></tr><tr><td><</td><td>Less than</td></tr><tr><td>></td><td>Greater than</td></tr><tr><td><=</td><td>Less than or equal to</td></tr><tr><td>>=</td><td>Greater than or equal to</td></tr><tr><td>= =</td><td>Equal to</td></tr><tr><td>!=</td><td>Not equal to</td></tr></table></td><td>2M (1/2 M each)
Any Four</td></tr>
<tr><td></td><td>b)</td><td>Enlist access specifiers in Java.</td><td>2 M</td></tr>
<tr><td></td><td>Ans</td><td>The access specifiers in java specify accessibility (scope) of a data member, method, constructor orclass. There are 5 types of java access specifier:
• public
• private</td><td>2M (1/2 M each)
Any Four</td></tr>
</table>

		• default (Friendly) • protected • private protected	
	c)	**Explain constructor with suitable example.**	**2 M**

	Ans	Constructors are used to assign initial value to instance variable of the class.
		It has the same name as class name in which it resides and it is syntactically similar to anymethod.
		Constructors do not have return value, not even 'void' because they return the instance if class.
		Constructor called by new operator.
		Example:
		class Rect
		{
		int length, breadth;
		Rect() //constructor
		{
		length=4; breadth=5;
		}
		public static void main(String args[])
		{
		Rect r = new Rect();
		System.out.println("Area : " +(r.length*r.breadth));
		}
		}
		Output : Area : 20
			1M- Explanation
			1M- Example

	d)	**List the types of inheritance which is supported by java.**	**2 M**
	Ans		**Any two** **1 M each**



	e)	**Define thread. Mention 2 ways to create thread.**	**2 M**

	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.
		<u>There are two ways to create threads in java:</u>
		1. By extending thread class
		Syntax: -
		class Mythread extends Thread
		{
		----- 	
		}
		2. Implementing the Runnable Interface
		Syntax:
		class MyThread implements Runnable
		{
		public void run()
		{
		------ 	
		}
			1 M- Define Thread
			
			1M -2ways to create thread

	f)	**Distinguish between Java applications and Java Applet (Any 2 points)**	**2 M**

Ans

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
Applets are event driven.	Applications are control driven.

1 M for each point (any 2 Points)

g) Draw the hierarchy of stream classes. **2 M**

Ans

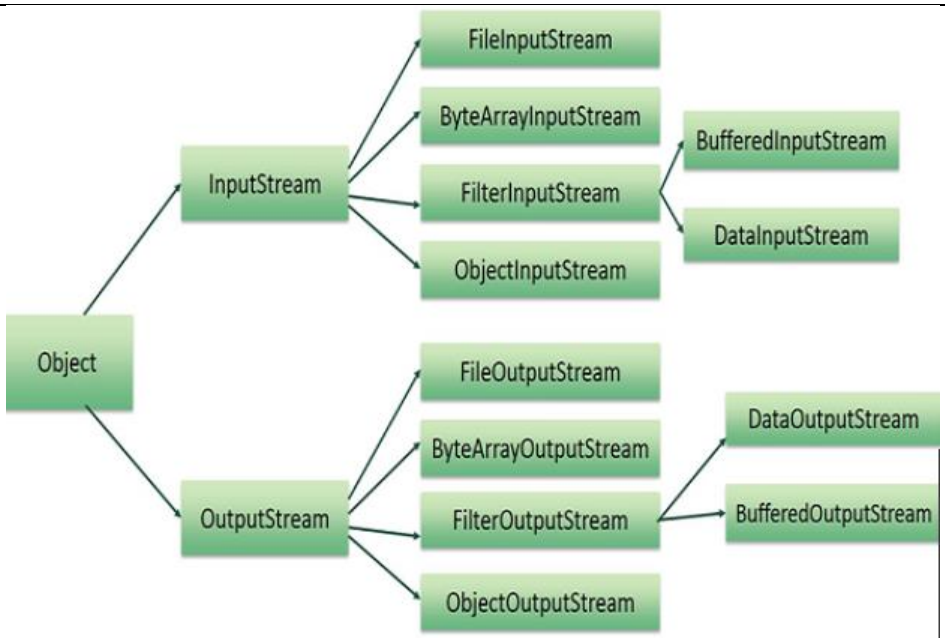


Fig: hierarchy of stream classes

2M-Correct diagram

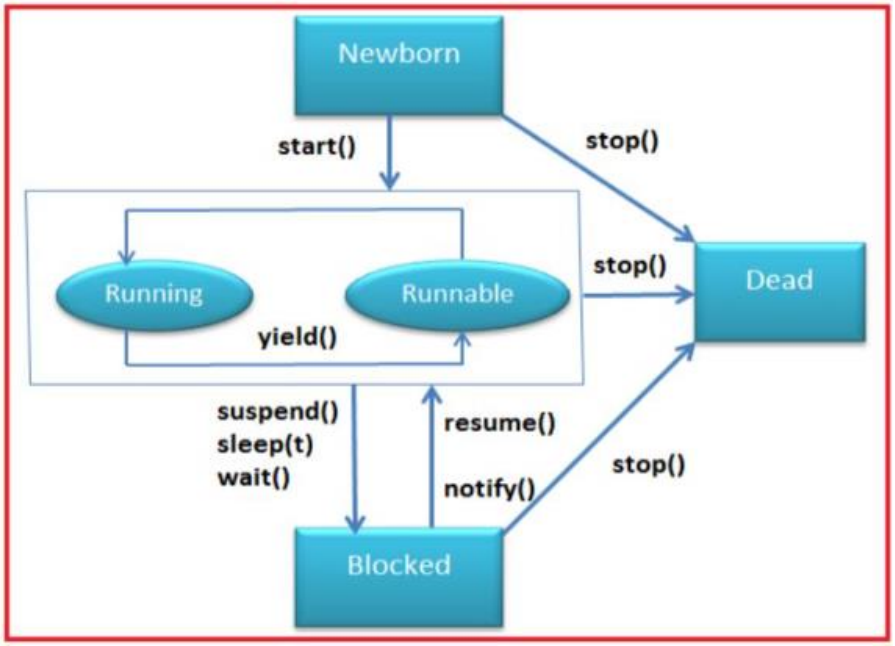
2.		**Attempt any THREE of the following:**	**12 M**
	a)	**Write a program to check whether the given number is prime or not.**	**4 M**

	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
			4M (for any correct program and logic)

	b)	**Define a class employee with data members 'empid , name and salary. Accept data for three objects and display it**	**4 M**

	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();
		}
		}
			4M (for correct program and logic)

	c)	**Describe Life cycle of thread with suitable diagram.**	**4 M**
	Ans	**1) Newborn State** A NEW Thread (or a Born Thread) is a thread that's been created but not yet started. It remains in this state until we start it using the start() method. The following code snippet shows a newly created thread that's in the NEW state: Runnable runnable = new NewState(); Thread t = new Thread(runnable); **2) Runnable State** It means that thread is ready for execution and is waiting for the availability of the processor i.e. the thread has joined the queue and is waiting for execution. 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 the control to another before its turn comes by yield(). Runnable runnable = new NewState(); Thread t = new Thread(runnable); t.start(); **3) 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. **4) 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. o suspend() : Thread can be suspended by this method. It can be rescheduled by resume(). o wait(): If a thread requires to wait until some event occurs, it can be done using wait method and can be scheduled to run again by notify(). o sleep(): We can put a thread to sleep for a specified time period using sleep(time) where time is in ms. It reenters the runnable state as soon as period has elapsed /over. **5) Dead State** Whenever we want to stop a thread form running further we can call its stop(). The stop() 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. The stop method may be used when the premature death is required	1M-digram of life cycle 3M-explanation

		 Fig: Life cycle of Thread	
	d)	**Write a program to read a file (Use character stream)**	**4 M**

	Ans	import java.io.FileWriter;
		import java.io.IOException;
		public class IOStreamsExample {
		public static void main(String args[]) throws IOException {
		//Creating FileReader object
		File file = new File("D:/myFile.txt");
		FileReader reader = new FileReader(file);
		char chars[] = new char[(int) file.length()];
		//Reading data from the file
		reader.read(chars);
		//Writing data to another file
		File out = new File("D:/CopyOfmyFile.txt");
		FileWriter writer = new FileWriter(out);
		//Writing data to the file
		writer.write(chars);
		writer.flush();
		System.out.println("Data successfully written in the specified file");
		}
		}
			4M (for correct program and logic)

3.		**Attempt any <u>THREE</u> of the following:**	**12 M**

	a)	**Write a program to find reverse of a number.**	**4 M**

	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);
		} }
			Any Correct program with proper logic -4M

	b)	**State the use of final keyword with respect to inheritance.**	**4 M**
	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.");	Use of final keyword-2 M Program-2 M

		}
		}
		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.	

	c)	**Give the usage of following methods** **i) drawPolygon ()** **ii) DrawOval ()** **iii) drawLine ()** **iv) drawArc ()**	**4 M**

	Ans	**i) drawPolygon ():**
		• drawPolygon() method is used to draw arbitrarily shaped figures.
		• Syntax: void drawPolygon(int x[], int y[], 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 xpoints[]={30,200,30,200,30};
		int ypoints[]={30,30,200,200,30};
		int num=5;
		g.drawPolygon(xpoints,ypoints,num);
		ii) drawOval ():
		• 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);
		ii) drawLine ():
		• The drawLine() method is used to draw line which take two pair of coordinates,
			Method use with description 1 M

(x1,y1) and (x2,y2) as arguments and draws a line between them.

- The graphics object g is passed to paint() method.
- Syntax: g.drawLine(x1,y1,x2,y2);
- Example: g.drawLine(100,100,300,300;)

iv) drawArc ()

drawArc( ) 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);

d) Write any four methods of file class with their use. **4 M**

Ans One method 1 M

public String getName()	Returns the name of the file or directory denoted by this abstract pathname.
public String getParent()	Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory
public String getPath()	Converts this abstract pathname into a pathname string.
public boolean isAbsolute()	Tests whether this abstract pathname is absolute. Returns true if this abstract pathname is absolute, false otherwise
public boolean exists()	Tests whether the file or directory denoted by this abstract pathname exists. Returns true if and only if the file or directory denoted by this abstract pathname exists; false otherwise
public boolean isDirectory()	Tests whether the file denoted by this abstract pathname is a directory. Returns true if and only if the file denoted by this abstract pathname exists and is a directory; false otherwise.
public boolean isFile()	Tests whether the file denoted by this abstract pathname is a normal file. A file is normal if it is not a directory and, in addition, satisfies other system-dependent criteria. Any nondirectory file created by a Java application is guaranteed to be a normal file. Returns true if and only if the file denoted by this abstract pathname exists and is a normal file; false otherwise.

4. Attempt any <u>THREE</u> of the following: **12 M**

a) Write all primitive data types available in Java with their storage Sizes in **4 M**

		bytes.	
	Ans	<table><tr><th>Data Type</th><th>Size</th></tr><tr><td>Byte</td><td>1 Byte</td></tr><tr><td>Short</td><td>2 Byte</td></tr><tr><td>Int</td><td>4 Byte</td></tr><tr><td>Long</td><td>8 Byte</td></tr><tr><td>Double</td><td>8 Byte</td></tr><tr><td>Float</td><td>4 Byte</td></tr><tr><td>Char</td><td>2 Byte</td></tr><tr><td>boolean</td><td>1 Bit</td></tr></table>	Data type name, size and default value and description carries 1 M
	b)	**Write a program to add 2 integer, 2 string and 2 float values in a vector. Remove the element specified by the user and display the list.**	**4 M**

	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);
		}
		}
			Correct program- 4 M, stepwise can give marks

	c)	**Develop a program to create a class ‘Book’ having data members author, title and price. Derive a class 'Booklnfo' having data member 'stock position’ and**	**4 M**

		method to initialize and display the information for three objects.	

	Ans	class Book
		{
		String author, title, publisher;
		Book(String a, String t, String p)
		{
		author = a;
		title = t;
		publisher = p;
		}
		}
		class BookInfo extends Book
		{
		float price;
		int stock_position;
		BookInfo(String a, String t, String p, float amt, int s)
		{
		super(a, t, p);
		price = amt;
		stock_position = s;
		}
		void show()
		{
		System.out.println("Book Details:");
		System.out.println("Title: " + title);
		System.out.println("Author: " + author);
		System.out.println("Publisher: " + publisher);
		System.out.println("Price: " + price);
		System.out.println("Stock Available: " + stock_position);
		}
		}
		class Exp6_1
		{
		public static void main(String[] args)
		{
		BookInfo ob1 = new BookInfo("Herbert Schildt", "Complete Reference", "ABC Publication", 359.50F,10);
		BookInfo ob2 = new BookInfo("Ulman", "system programming", "XYZ Publication", 359.50F, 20);
		BookInfo ob3 = new BookInfo("Pressman", "Software Engg", "Pearson Publication", 879.50F, 15);
		ob1.show();
			Correct program- 4 M

		ob2.show();
		ob3.show();
		}
		}
		OUTPUT
		Book Details:
		Title: Complete Reference
		Author: Herbert Schildt
		Publisher: ABC Publication
		Price: 2359.5
		Stock Available: 10
		Book Details:
		Title: system programming
		Author: Ulman
		Publisher: XYZ Publication
		Price: 359.5
		Stock Available: 20
		Book Details:
		Title: Software Engg
		Author: Pressman
		Publisher: Pearson Publication
		Price: 879.5
		Stock Available: 15	

	d)	**Mention the steps to add applet to HTML file. Give sample code.**	**4 M**
	Ans	Adding Applet to the HTML file: Steps to add an applet in HTML document 1. Insert an <APPLET> tag at an appropriate place in the web page i.e. in the body section of HTML file. 2. Specify the name of the applet's .class file. 3. If the .class file is not in the current directory then use the codebase parameter to specify:- a. the relative path if file is on the local system, or b. the uniform resource locator(URL) of the directory containing the file if it is on a remote computer. 4. Specify the space required for display of the applet in terms of width and height in pixels. 5. Add any user-defined parameters using <param> tags 6. Add alternate HTML text to be displayed when a non-java browser is used. 7. Close the applet declaration with the </APPLET> tag. Open notepad and type the following source code and save it into file name	Steps – 2M Example – 2 M

"Hellojava.java"

```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet
{
 public void paint (Graphics g)
{
g.drawString("Hello Java",10,100);
} }
```

Use the java compiler to compile the applet "Hellojava.java" file.

C:\jdk> javac Hellojava.java

After compilation "Hellojava.class" file will be created. Executable applet is nothing but the .class file of the applet, which is obtained by compiling the source code of the applet. If any error message is received, then check the errors, correct them and compile the applet again.

We must have the following files in our current directory.

- o Hellojava.java
- o Hellojava.class
- o HelloJava.html

If we use a java enabled web browser, we will be able to see the entire web page containing the applet.

We have included a pair of <APPLET..> and </APPLET> tags in the HTML body section. The <APPLET…> tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires. The <APPLET> tag given below specifies the minimum requirements to place the HelloJava applet on a web page. The display area for the applet output as 300 pixels width and 200 pixels height. CENTER tags are used to display area in the center of the screen.

<APPLET CODE = hellojava.class WIDTH = 400 HEIGHT = 200 > </APPLET>

Example: Adding applet to HTML file:

Create Hellojava.html file with following code:

```
<HTML>
<! This page includes welcome title in the title bar and displays a welcome message. Then it specifies
the applet to be loaded and executed.
>
<HEAD> <TITLE> Welcome to Java Applet </TITLE> </HEAD>
<BODY> <CENTER> <H1> Welcome to the world of Applets </H1> </CENTER> <BR>
<CENTER>
```

		<APPLET CODE=HelloJava.class WIDTH = 400 HEIGHT = 200 > </APPLET> </CENTER> </BODY> </HTML>	
	e)	**Write a program to copy contents of one file to another.**	**4 M**

	Ans	import java.io.*;
		class copyf
		{
		public static void main(String args[]) throws IOException
		{
		BufferedReader in=null;
		BufferedWriter out=null;
		try
		{
		in=new BufferedReader(new FileReader("input.txt"));
		out=new BufferedWriter(new FileWriter("output.txt"));
		int c;
		while((c=in.read())!=-1)
		{
		out.write(c);
		}
		System.out.println("File copied successfully");
		}
		finally
		{
		if(in!=null)
		{
		in.close();
		}
		if(out!=null)
		{
		out.close();
		}
		}
		}
			Correct program- 4 M

5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Compare array and vector. Explain elementAT( ) and addElement( ) methods.**	**6 M**
	Ans		

Sr. No.	Array	Vector
1	An array is a structure that holds multiple values of the same type.	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.	Vectors are heterogeneous. You can have objects of different data types inside a Vector.
3	After creation, an array is a fixed-length structure.	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.	Vector are store non-primitive type data element
5	Array is unsynchronized i.e. automatically increase the size when the initialized size will be exceed.	Vector is synchronized i.e. when the size will be exceeding at the time; vector size will increase double of initial size.
6	Declaration of an array : int arr[] = new int [10];	Declaration of Vector: Vector list = new Vector(3);
7	Array is the static memory allocation.	Vector is the dynamic memory allocation
8	Array allocates the memory for the fixed size ,in array there is wastage of memory.	Vector allocates the memory dynamically means according to the requirement no wastage of memory.
9	No methods are provided for adding and removing elements.	Vector provides methods for adding and removing elements.
10	In array wrapper classes are not used.	Wrapper classes are used in vector
11	Array is not a class.	Vector is a class.

4 M for any 4 correct points

1 M for elementAt()

1 M for addElement ()

elementAT( ):

The **elementAt()** method of Java Vector class is used to get the element at the specified

		index in the vector. Or The **elementAt()** method returns an element at the specified index. addElement( ): The **addElement()** method of **Java Vector** class is used to add the specified element to the end of this vector. Adding an element increases the vector size by one.	
	b)	**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.**	**6 M**

	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)
		{
			6 M for correct program

```
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();
 }
 }
```

c) **Define an exception called 'No Match Exception' that is thrown when the password accepted is not equal to "MSBTE'. Write the program.** **6 M**

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
{
```

6 M for correct program

```
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());
}
}
}
```

6. **Attempt any TWO of the following:** **12 M**

a) **Write a program to check whether the string provided by the user is palindrome or not.** **6 M**

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)
{
```

6 M for correct program

```
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");
 }
}
 }
```

	b)	**Define thread priority ? Write default priority values and the methods to set and change them.**	**6 M**
	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 the java scheduler. Default priority values as follows	2 M for define Thread priority 2 M for

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 the given 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 is 1 (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());
}
```

default priority values

2 M for method to set and change

		}	
	c)	**Design an applet to perform all arithmetic operations and display the result by using labels. textboxes and buttons.**	**6 M**

	Ans	**import** java.awt.*;
		import java.awt.event.*;
		public class sample **extends** Frame **implements** ActionListener {
		Label l1, l2,l3;
		TextField tf1, tf2, tf3;
		Button b1, b2, b3, b4;
		sample() {
		l1=new Lable(“First No.”);
		l1.setBounds(10, 10, 50, 20);
		tf1 = **new** TextField();
		tf1.setBounds(50, 50, 150, 20);
		l2=new Lable(“Second No.”);
		l2.setBounds(10, 60, 50, 20);
		tf2 = **new** TextField();
		tf2.setBounds(50, 100, 150, 20);
		l3=new Lable(“Result”);
		l3.setBounds(10, 110, 150, 20);
		tf3 = **new** TextField();
		tf3.setBounds(50, 150, 150, 20);
		tf3.setEditable(**false**);
		b1 = **new** Button("+");
		b1.setBounds(50, 200, 50, 50);
		b2 = **new** Button("-");
		b2.setBounds(120,200,50,50);
		b3 = **new** Button("*");
		b3.setBounds(220, 200, 50, 50);
		b4 = **new** Button("/");
		b4.setBounds(320,200,50,50);
		b1.addActionListener(**this**);
		b2.addActionListener(**this**);
		b3.addActionListener(**this**);
		b4.addActionListener(**this**);
		
		add(tf1);
		add(tf2);
		add(tf3);
		add(b1);
		add(b2);
		add(b3);
		add(b4);
			6 M for correct program

```
            setSize(400,400);
            setLayout(null);
            setVisible(true);
        }
        public void actionPerformed(ActionEvent e) {
String s1 = tf1.getText();
            String s2 = tf2.getText();
            int a = Integer.parseInt(s1);
            int b = Integer.parseInt(s2);
            int c = 0;
            if (e.getSource() == b1){
               c = a + b;
            }
            else if (e.getSource() == b2){
               c = a - b;
            else if (e.getSource() == b3){
               c = a * b;
            else if (e.getSource() == b4){
               c = a / b;
            }
            String result = String.valueOf(c);
tf3.setText(result);
        }
    public static void main(String[] args) {
       new sample();
    }
    }
```