V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

+91 93260 50669

V2V EdTech LLP

v2vedtech.com

v2vedtech

SUMMER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Java Programming **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.

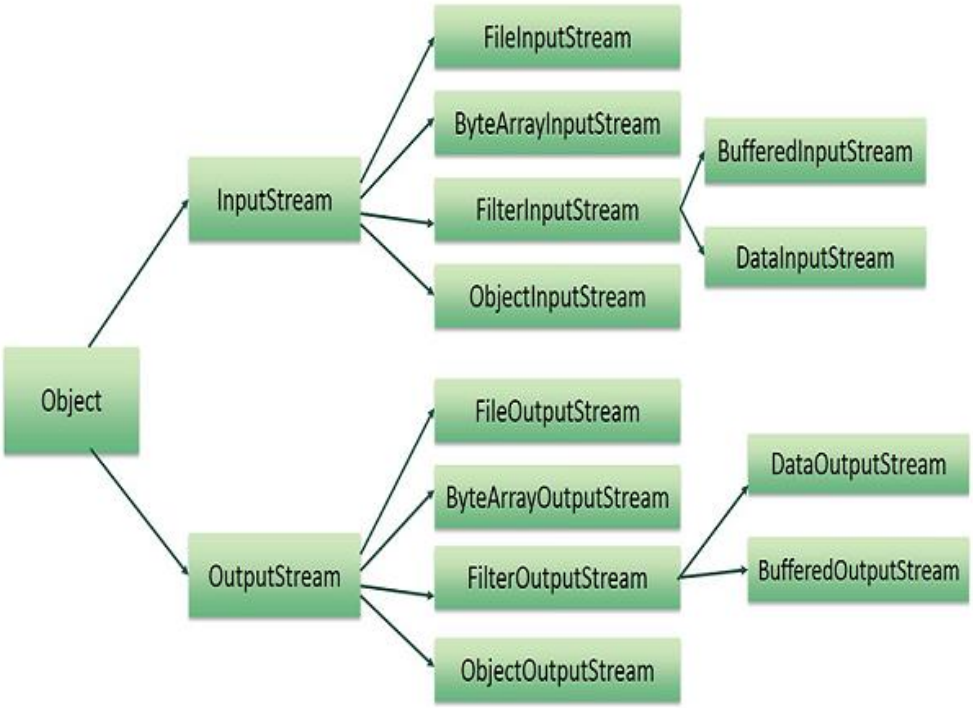
Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**Define the terms with example.** **i) Class** **ii) Object**	**2 M**
	Ans	**i) Class:** Class is a set of object, which shares common characteristics/ behavior and common properties/ attributes. **ii) Object**: It is a basic unit of Object-Oriented Programming and represents real-life entities. Example: **class** Student { **int** id; String name;	1 M for any suitable class definition and 1 M for any suitable object definition

		public static void main(String args[])
		{
		Student s1=**new** Student(); //creating an object of Student
		}
		}
		
		In this example, we have created a Student class which has two data members id and name. We are creating the object of the Student s1 by new keyword.	

	b)	**Enlist any two access specifier with syntax.**	**2 M**
	Ans	There are 5 types of java access specifier: • public • private • default (Friendly) • protected • private protected	List any 2 access specifiers - 2M
	c)	**Give a syntax to create a package and accessing package in java.**	**2 M**
	Ans	**<u>To Create a package follow the steps given below:</u>** • Choose the name of the package • Include the package command as the first line of code in your Java Source File. • The Source file contains the classes, interfaces, etc. you want to include in the package • Compile to create the Java packages **<u>Syntax to create a package:</u>** package nameOfPackage; Example: package p1; **<u>Accessing Package:</u>** • Package can be accessed using keyword import. • There are 2 ways to access java system packages: o Package can be imported using import keyword and the wild card(*) but drawback of this shortcut approach is that it is difficult to determine from which package a particular member name. Syntax: import package_name.*;	syntax to create a package-1 M and accessing package-1 M

		For e.g. import java.lang.*; o The package can be accessed by using dot(.) operator and can be terminated using semicolon(;) Syntax: import package1.package2.classname;	
	d)	**Give a syntax of following thread method** **i) Notify ( )** **ii) Sleep ( )**	**2 M**
	Ans	<u>i) notify()</u> The **notify()** method of thread class is used to wake up a single thread. This method gives the notification **for only one thread** which is waiting for a particular object. Syntax: **public final void** notify() <u>ii) sleep()</u> Sleep() causes the current thread to suspend execution for a specified period. Syntax: public static void sleep(long milliseconds)	Syntax of notify( )-1 M and sleep ()-1 M
	e)	**Give a syntax of (param) tag to pass parameters to an applet.**	**2 M**

	Ans	User-define Parameter can be applied in applet using <PARAM…> tags. Each <PARAM…> tag has a name and value attribute.
		Syntax: <PARAM name = …… Value = "……" >
		For example, the param tags for passing name and age parameters looks as shown below:
		<param name="name" value="Ramesh" />
		<param name="age" value="25″ />
		The *getParameter()* method of the *Applet* class can be used to retrieve the parameters passed from the HTML page. The syntax of *getParameter()* method is as follows:
		String getParameter(String param-name);
		Example:
		public void init()
		{
		n = getParameter("name");
		a = getParameter("age");
		}
			Syntax of <param> - 1 M
			And syntax of getParameter ( )- 1 M

	f)	**Define stream class and list types of stream class.**	**2 M**
	Ans	A java stream is a group of objects that can be piped together to produce the desired result. Streams are used in Java to transfer data between programs and I/O devices like a file, network connections, or consoles.  Fig: Types of stream classes	Define stream class=1 M and any 2 types of stream class=1 M
	g)	**Give use of garbage collection in java.**	**2 M**
	Ans	The garbage collector provides the following uses: 1. Frees developers from having to manually release memory means destroy the unused objects 2. Allocates objects on the managed heap efficiently. 3. Reclaims objects that are no longer being used, clears their memory, and keeps the memory available for future allocations. 4. It is automatically done by the garbage collector(a part of JVM), so we don't need extra effort.	Any 2 uses=2 M
2.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Describe type casting in java with example.**	**4 M**
	Ans	In Java, **type casting** is a method or process that converts a data type into another data	Definition of

type in both ways manually and automatically. The automatic conversion is done by the compiler and manual conversion performed by the programmer.

Type casting is of two types: widening, narrowing.

Widening (Implicit)

- The process of assigning a smaller type to a larger one is known as widening or implicit.

  Byte → short → int → long → float → double

For e.g.

```
class widening
{
public static void main(String arg[])
{
int i=100;
long l=I;
float f=l;
System.out.println("Int value is"+i);
System.out.println("Long value is"+l);
System.out.println("Float value is"+f);
}
}
```

Narrowing (Explicit)

- The process of assigning a larger type into a smaller one is called narrowing.
- Casting into a smaller type may result in loss of data.
- double → long → int → short → byte

For e.g.

```
class narrowing
{
Public static void main(String[])
{
Double d=100.04;
Long l=(long) d;
Int i=(int) l;
```

type casting-1 M

Types of type casting-1 M

Example-2 M

(1 M for each example)

```
                System.out.println("Int value is"+i);
                System.out.println("Long value is"+l);
                System.out.println("Float value is"
                }
}
```

b) Differentiate between String and String Buffer Class. (any four points) — **4 M**

Ans

String class	**StringBuffer class**
String is a major class	StringBuffer 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 ofStringBuffer class using "new"
Methods of string class: toLowerCase(), toUpperCase(), replace(), trim(), equals(), length(), charAt(), concat(), substring(), compareTo()	Methods of StringBuffer class: setCharAt(), append(), insert(), append(), reverse(), delete()
Ex:- String s="abc"‖;	Ex:- StringBuffer s=new StringBuffer ("abc");

Any 4 correct points-4 M

c) Write a program to create a user defined exception in java. — **4 M**

Ans **Following example shows a user defined exception as 'Invalid Age', if age entered by the user is less than eighteen.**

For any Correct program-4 M

```
import java.lang.Exception;
import java.io.*;
class myException extends Exception
{
myException(String msg)
{
super(msg);
}
}
class agetest
{
public static void main(String args[])
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
//Scanner class is also valid
try
{
System.out.println("enter the age : ");
int n=Integer.parseInt(br.readLine());
if(n < 18 )
throw new myException("Invalid Age"); //user defined exception
```

		else
		System.out.println("Valid age");
		}
		catch(myException e)
		{
		System.out.println(e.getMessage());
		}
		catch(IOException ie)
		{}
		}
		}	

	d)	**Write a program for reading and writing character to and from the given files using character stream classes.**	**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");
		}
		}
			4 M (for any correct program and logic)

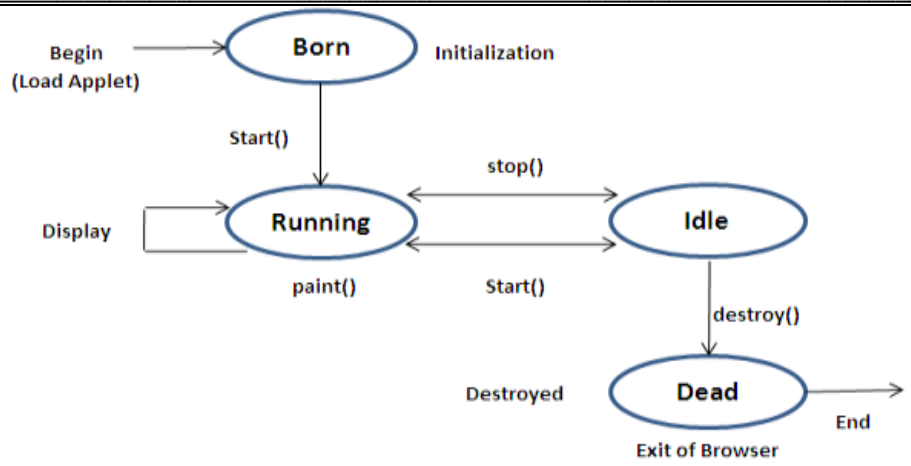
3.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Write a program to print all the Armstrong numbers from 0 to 999**	**4 M**

	Ans	import java.util.Scanner;
		class ArmstrongWhile
		{
		public static void main(String[] arg)
		{
		int i=0,arm;
		System.out.println("Armstrong numbers between 0 to 999");
		while(i<1000)
			Correct logic – 4 M

```
        {
        arm=armstrongOrNot(i);
        if(arm==i)
        System.out.println(i);
        i++;
        }
        }
static int armstrongOrNot(int num)
{
        int x,a=0;
        while(num!=0)
        {
                x=num%10;
                a=a+(x*x*x);
                num/=10 ;
        }
        return a;
}
}

OR
class ArmstrongWhile
{
        public static void main(String[] arg)
        {
        int i=1,a,arm,n,temp;
        System.out.println("Armstrong numbers between 0 to 999 are");
        while(i<500)
        {
        n=i;
        arm=0;
        while(n>0)
        {
                a=n%10;
                arm=arm+(a*a*a);
                n=n/10;
        }
        if(arm==i)
                System.out.println(i);
        i++;
        }
        }
}
```

	b)	**Explain the applet life cycle with neat diagram.**	**4 M**

	Ans	
		Applet Life Cycle: An Applet has a life cycle, which describes how it starts, how it operates and how it ends. The life cycle consists of four methods: init(), start(), stop() and destroy().
		Initialization State (The init() method):
		The life cycle of an Applet is begin on that time when the applet is first loaded into the browser and called the init() method. The init() method is called only one time in the life cycle on an Applet. The init() method is basically called to read the "PARAM" tag in the html file. The init () method retrieve the passed parameter through the "PARAM" tag of html file using get Parameter() method All the initialization such as initialization of variables and the objects like image, sound file are loaded in the init () method .After the initialization of the init() method user can interact with the Applet and mostly applet contains the init() method.
		We may do following thing if required.
		• Create objects needed by the applet
		• Set up initial values
		• Load images or fonts
		• Set up colors
		Running State (The start() method): The start method of an Applet is called after the initialization method init(). This method may be called multiples time when the Applet needs to be started or restarted. For Example if the user wants to return to the Applet, in this situation the start() method of an Applet will be called by the web browser and the user will be back on the applet. In the start method user can interact within the applet.
		public void start()
		{
		……..
		……..
		}
		Idle (The Stop() method): An applet becomes idle when it is stopped from running. The stop() method stops the applet and makes it invisible. Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the stop() method explicitly. The stop() method can be called multiple times in the life cycle of applet like the start () method or should be called at least one time. For example the stop() method is called by the web browser on that time When the user leaves one applet to go another applet and the start() method is called on that time when the user wants to go back into the first program or Applet.
		public void stop()
			Diagram-2 M and explanation – 2 M

		{
		……..
		……..
		}
		Dead State (The destroy() method): The destroy() method is called to terminate an Applet. An Applet is said to be dead when it is removed from memory. This occurs automatically by invoking the destroy() method when we quit the browser. It is useful for clean-up actions, such as releasing memory after the applet is removed, killing off threads and closing network/database connections.
		Thus this method releases all the resources that were initialized during an applet's initialization.
		public void destroy()
		{
		……..
		……..
		}
		Display State (The paint() method): The paint() method is used for applet display on the screen.
		The display includes text, images, graphics and background. This happens immediately after the applet enters into the running state. Almost every applet will have a paint() method and can be called several times during an applet's life cycle. The paint() method is called whenever a window is required to paint or repaint the applet.
		public void paint(Graphics g)
		{
		……..
		……..
		}	

	c)	**Describe the package in java with suitable example.**	**4 M**
	Ans	• Java provides a mechanism for partitioning the class namespace into more manageable parts called package (i.e package are container for a classes). • 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. Syntax: package pkg; pkg is the name of the package eg : package mypack; • 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. • The classes and methods of a package must be public. • 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|*);	Description- 2 M, Example -2 M

		• Example:
		package1:
		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 VolumeDemo
		{
		public static void main(String args[])
		{
		Box b=new Box();
		b.display();
		}
		}	

	d)	**Enlist types of Byte stream class and describe input stream class and output stream class.**	**4 M**
	Ans	• Byte streams class: It handles I/O operations on bytes. • InputStream and OutputStream classes are operated on bytes for reading and writing, respectively. • Byte streams are used in a program to read and write 8-bit bytes. • InputStream and OutputStream are the abstract super classes of all byte streams that have a sequential nature. • The stream is unidirectional; they can transmit bytes in only one direction. • InputStream and OutputStream provide the Application program Interface (API) and partial implementation for input streams (streams that read bytes) and output streams (streams that write bytes). • Input Stream Classes: java.io.InputStream is an abstract class that contains the basic methods for reading raw bytes of data from a stream. The InputStream class defines methods for performing the input functions like: reading bytes, closing streams, marking positions in streams, skipping ahead in a stream and finding the number of bytes in a stream. • **Input stream class methods:** 1. int read ()- Returns an integer representation of next available byte of input.-1 is returned at the stream end. 2. int read (byte buffer[ ])- Read up to buffer.length bytes into buffer & returns actual number	Type – 1 M, Explanation -3 M

		of bytes that are read. At the end returns –1. 3. int read(byte buffer[ ], int offset, int numbytes)- Attempts to read up to numbytes bytes into buffer starting at buffer[offset]. Returns actual number of bytes that are read. At the end returns –1. 4. void close()- to close the input stream 5. void mark(int numbytes)- places a mark at current point in input stream and remain valid till number of bytes are read. 6. void reset()- Resets pointer to previously set mark/ goes back to stream beginning. 7. long skip(long numbytes)- skips number of bytes. 8. int available()- Returns number of bytes currently available for reading. • **Output Stream Classes:** • The java.io.OutputStream class sends raw bytes of data to a target such as the console or a network server. Like InputStream, OutputStream is an abstract class. • The OutputStream includes methods that perform operations like: writing bytes, closing streams,flushing streams etc. • Methods defines by the OutputStream class are 1. void close() - to close the OutputStream 2. void write (int b) - Writes a single byte to an output stream. 3. void write(byte buffer[ ]) - Writes a complete array of bytes to an output stream. 4. void write (byte buffer[ ], int offset, int numbytes) - Writes a sub range of numbytes bytes from the array buffer, beginning at buffer[offset]. 5. void flush() - clears the buffer.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Describe any four features of java.**	**4 M**
	Ans	**1. Compile & Interpreted:** Java is a two staged system. It combines both approaches. First java compiler translates source code into byte code instruction. Byte codes are not machine instructions. In the second stage java interpreter generates machine code that can be directly executed by machine. Thus java is both compile and interpreted language. **2. Platform independent and portable:** Java programs are portable i.e. it can be easily moved from one computer system to another. Changes in OS, Processor, system resources won't force any change in java programs. Java compiler generates byte code instructions that can be implemented on any machine as well as the size of primitive data type is machine independent. **3. Object Oriented:** Almost everything in java is in the form of object. All program codes and data reside within objects and classes. Similar to other OOP languages java also has basic OOP properties such as encapsulation, polymorphism, data abstraction, inheritance etc. Java comes with an extensive set of classes (default) in packages. **4. Robust & Secure:** Java is a robust in the sense that it provides many safeguards to ensure reliable codes. Java incorporates concept of exception handling which captures errors and eliminates any risk of crashing the system. Java system not only verify all memory access but also ensure that no viruses are communicated with an applet. It does not use pointers by which you can gain access to memory locations without proper	Any four each features -1 M each

authorization.

5. Distributed: It is designed as a distributed language for creating applications on network. It has ability to share both data and program. Java application can open and access remote object on internet as easily as they can do in local system.

6. Multithreaded: It can handle multiple tasks simultaneously. Java makes this possible with the feature of multithreading. This means that we need not wait for the application to finish one task before beginning other.

7. Dynamic and Extensible: Java is capable of dynamically linking new class library's method and object. Java program supports function written in other languages such as C, C++ which are called as native methods. Native methods are linked dynamically at run time.

b) Explain any four methods of vector class with example. **4 M**

Ans 1 Method – 1 M

Vector class is in java.util package of java.
Vector is dynamic array which can grow automatically according to the requirement.
Vector does not require any fix dimension like String array and int array.
Vectors are used to store objects that do not have to be homogeneous.
Vector contains many useful methods.
Vectors are created like arrays. It has three constructor methods

- Vector list = new Vector(); //declaring vector without size
- Vector list = new Vector(3); //declaring vector with size
- Vector list = new Vector(5,2); //create vector with initial size and whenever it need to grows, it grows by value specified by increment capacity.

Method Name	**Task performed**
list.firstElement()	It returns the first element of the vector.
list.lastElement()	It returns last element of the vector
list.addElement(item)	Adds the item specified to the list at the end.
list.elementAt(n)	Gives the name of the object at nth position
list.size()	Gives the number of objects present in vector
List.capacity()	This method returns the current capacity of the vector.
list.removeElement(item)	Removes the specified item from the list.
list.removeElementAt(n)	Removes the item stored in the nth position of the list.
list.removeAllElements()	Removes all the elements in the list.
list.insertElementAt(item, n)	Inserts the item at nth position.
List.contains(object element)	This method checks whether the specified element is present inthe Vector. If the element is been found it returns true else false.
list.copyInto(array)	Copies all items from list of array.

Example:

```
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");
Character s5=new
Character('a');Character
s6=new Character('b');
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(s5);
v.addElement(s6);
v.addElement(s7);
v.addElement(s8);
System.out.println(v);
v.removeElement(s2);
v.removeElementAt(4);
System.out.println(v);
}
}
```

	c)	**Describe interface in java with suitable example.**	**4 M**
	Ans	Java does not support multiple inheritances with only classes. Java provides an alternate approach known as interface to support concept of multiple inheritance. An interface is similar to class which can define only abstract methods and final variables. **Syntax:** access interface InterfaceName { Variables declaration; Methods declaration;	Interface explanation – 2 M, any suitable example – 2 M

```
}
```

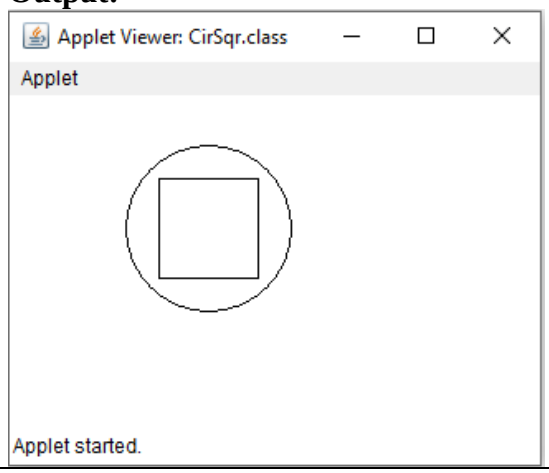
Example:

```
interface sports
{
int sport_wt=5;
public void disp();
}
class test
{
int roll_no;
String name;
int m1,m2;
test(int r, String nm, int m11,int m12)
{
roll_no=r;
name=nm;
m1=m11;
m2=m12;
}
}
class result extends test implements sports
{
result (int r, String nm, int m11,int m12)
{
super (r,nm,m11,m12);
}
public void disp()
{
System.out.println("Roll no : "+roll_no);
System.out.println("Name : "+name);
System.out.println("sub1 : "+m1);
System.out.println("sub2 : "+m2);
System.out.println("sport_wt : "+sport_wt);
int t=m1+m2+sport_wt;
System.out.println("total : "+t);
}
public static void main(String args[])
{
result r= new result(101,"abc",75,75);
r.disp();
}
}
```

Output :

```
D:\>java result
Roll no : 101
Name : abc
sub1 : 75
sub2 : 75
sport_wt : 5
```

		total : 155	
	d)	**Write an applet program for following graphics method.** **i) Drawoval ( )** **ii) Drawline ( )**	**4 M**

	Ans	import java.awt.*;
		import java.applet.*;
		public class CirSqr extends Applet
		{
		public void paint(Graphics g)
		{
		g.drawOval(70,30,100,100);
		g.drawRect(90,50,60,60);
		}
		}
		/*<applet code="CirSqr.class" height=200 width=200>
		</applet> */
		Output:
		 
		
			Any correct logic can be considered 2 M for drawoval and 2 M for drawline

	e)	**Enlist any four methods of file input stream class and give syntax of any two methods.**	**4 M**
	Ans	• Input Stream Classes: java.io.InputStream is an abstract class that contains the basic methods for reading raw bytes of data from a stream. The InputStream class defines methods for performing the input functions like: reading bytes, closing streams, marking positions in streams, skipping ahead in a stream and finding the number of bytes in a stream. • Input stream class methods: 1. int read ()- Returns an integer representation of next available byte of input.-1 is returned at the stream end. 2. int read (byte buffer[ ])- Read up to buffer.length bytes into buffer & returns actual number of bytes that are read. At the end returns –1. 3. int read(byte buffer[ ], int offset, int numbytes)- Attempts to read up to numbytes bytes into buffer starting at buffer[offset]. Returns actual number of bytes that are read.	One method – 1 M

		At the end returns –1. 4. void close()- to close the input stream 5. void mark(int numbytes)- places a mark at current point in input stream and remain valid till number of bytes are read. 6. void reset()- Resets pointer to previously set mark/ goes back to stream beginning. 7. long skip(long numbytes)- skips number of bytes. 8. int available()- Returns number of bytes currently available for reading.	
5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Write a program to copy all elements of one array into another array.**	**6 M**

	Ans	public class CopyArray {
		public static void main(String[] args)
		{
		int [] arr1 = new int [] {1, 2, 3, 4, 5};
		
		int arr2[] = new int[arr1.length];
		
		for (int i = 0; i < arr1.length; i++)
		{
		arr2[i] = arr1[i];
		}
		System.out.println("Elements of original array: ");
		
		for (int i = 0; i < arr1.length; i++)
		{
		System.out.print(arr1[i] + " ");
		}
		
		System.out.println();
		
		System.out.println("Elements of new array: ");
		
		for (int i = 0; i < arr2.length; i++)
		{
		System.out.print(arr2[i] + " ");
			6 M for any correct program and logic

```
            }
        }
}
```

b) **Write a program to implement the following inheritance. Refer Fig. No. 1.** **6 M**

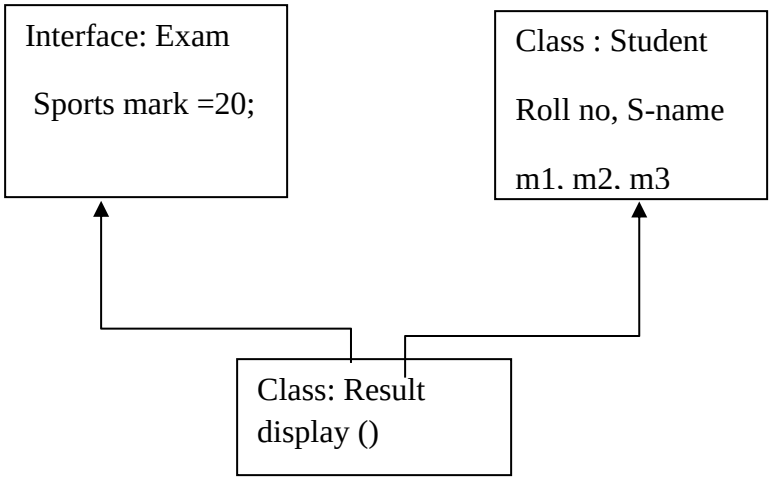


Fig. No. 1

Ans 6 M for correct program

```
interface Exam
{
 int sports_mark=20;
}
class Student
{
        String S-name;
        int Roll_no, m1, m2, m3;
        Student(String n, int a, int b, int c, int d)
{
S-name = n;
Roll_no = a;
```

```
m1 = b;
m2 = c;
m3 = d;
}
void showdata()
{
System.out.println("Name of student :"+S-name);
System.out.println("Roll no. of the students :"+Roll_no);
System.out.println("Marks of subject 1:"+m1);
System.out.println("Marks of subject 2:"+m2);
System.out.println("Marks of subject 3:"+m3);
}
}
class Result extends Student implements Exam
{
Result(String n, int a, int b, int c, int d)
{
super(n, a, b, c, d );
}
void dispaly()
{
super.showdata();
int total=(m1+m2+m3);
float result=(total+Sports_mark)/total*100;
```

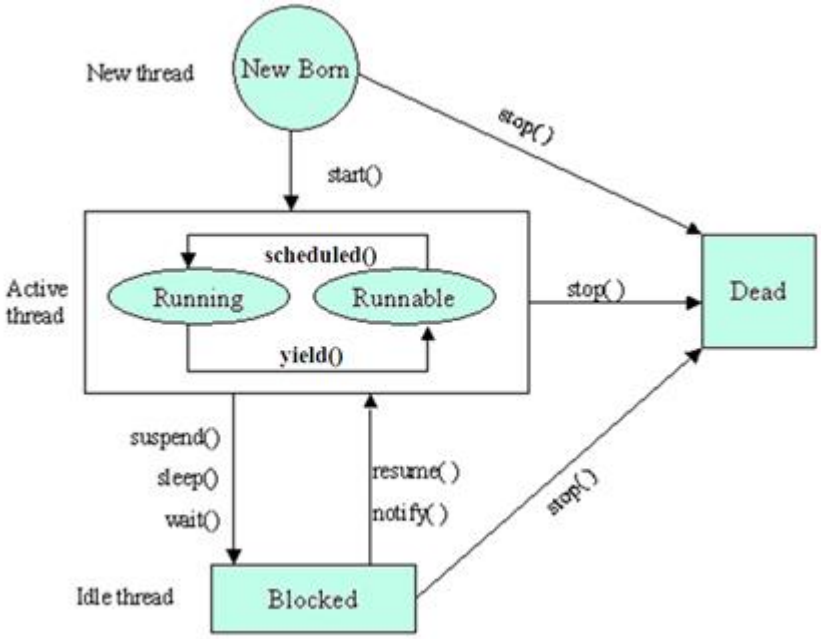
		System.out.println("result of student is:"+result);
		}
		}
		class studentsDetails
		{
		public static void main(String args[])
		{
		Result r=new Result("Sachin",14, 78, 85, 97);
		r.display();
		}
		}	

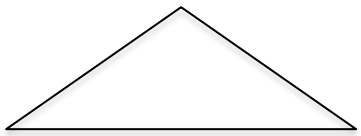
	c)	**Write a program to print even and odd number using two threads with delay of 1000ms after each number.**	**6 M**

	Ans	class odd extends Thread
		{
		public void run()
		{
		for(int i=1;i<=20;i=i+2)
		{
		System.out.println("ODD="+i);
		try
		{
		sleep(1000);
		}
		catch(Exception e)
		{
		System.out.println("Error");
			6 M for correct program

```
}
}
}
}
class even extends Thread
{
public void run()
{
for(int i=0;i<=20;i=i+2)
{
System.out.println("EVEN="+i);
try
{
sleep(1000);
}
catch(Exception e)
{
System.out.println("Error");
}
}
}
}
class oddeven
{
public static void main(String arg[])
{
odd o=new odd();
even e=new even();
```

```
o.start();
e.start();
}
}
```

6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Explain thread life cycle with neat diagram.**	**6 M**
	Ans	 Thread Life Cycle Thread has five different states throughout its life. **1) 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 it can be scheduled for running by start() or killed by stop(). If put in a queue it moves to runnable state. **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().	2 M for diagram, 4 M for explanation

		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 Thread should be in any one state of above and it can be move from one state to another by different methods and ways.	
	b)	**Write a program to generate following output using drawline () method. Refer Fig. No. 2.**  **Fig. No. 2**	**6 M**

	Ans	**Using drawLine() method**
		import java.applet.*;
		import java.awt.*;
		public class Triangle extends Applet
		{
		public void paint(Graphics g)
		{
		g.drawLine(100,200,200,100);
			6 M for correct program

		g.drawLine(200,100,300,200);
		g.drawLine(300,200,100,200);
		}
		}
		/*<applet code="Triangle.class" height=300 width=200> </applet>*/
		OR
		Using drawPolygon() method
		import java.applet.*;
		import java.awt.*;
		public class Triangle extends Applet
		{
		public void paint(Graphics g)
		{
		int a[]={100,200,300,100};
		int b[]={200,100,200,200};
		int n=4;
		g.drawPolygon(a,b,n);
		}
		}
		/*<applet code="Triangle.class" height=300 width=200> </applet>*/	

	c)	**Explain constructor with its type. Give an example of parameterized constructor.**	**6 M**
	Ans	**Constructor:** A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes. Types of constructors are: **Default constructor :** It is constructor which is inserted by Java compiler when no constructor is provided in class. Every class has constructor within it. Even abstract class have default constructor. By default, Java compiler, insert the code for a zero parameter constructor.	2 M for definition of constructor and types of constructors 4 M for example (for any correct suitable program of parameterize

		Default constructor is the no arguments constructor automatically generated unless you define another constructor.
		The default constructor automatically initializes all numeric members to zero and other types to null or spaces.
		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));
		}
		}
		Parameterized constructor: Constructor which have arguments are known as parameterized constructor.
		When constructor method is defined with parameters inside it, different value sets can be provided to different constructor with the same name.
		Example of Parameterized Constructor
		class Rect
		{
		int length, breadth;
		Rect(int l, int b) // parameterized constructor
		{
		length=l;
		breadth=b;
		}
		public static void main(String args[])
		{
		Rect r = new Rect(4,5); // constructor with parameters
		Rect r1 = new Rect(6,7);
		System.out.println("Area : " +(r.length*r.breadth));
			d constructor)

		System.out.println("Area : " +(r1.length*r1.breadth)); } }	