Important Instructions to examiners:

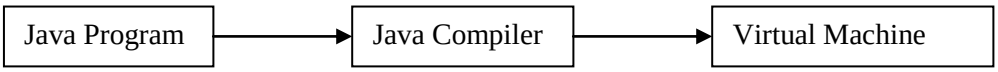
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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.

Q.1.

A. Attempt any THREE of the following: 12

a) What is JVM? What is byte code?
(JVM- 3 M, bytecode- 1 M)

JVM is the Java Virtual Machine
The java compiler compiles produces an intermediate code known as byte code for the java virtual machine, which exists only in the computer memory
It is a simulated computer within the computer and does all the major functions of a real computer



Source code
Byte Code

process of compilation

Virtual machine code is not machine specific
Machine specific code is generated by Java Interpreter by acting as an intermediary between the virtual machine and the real machine.
Interpreter is written for each type of machine.

Virtual machine
real machine

Process of converting byte code into machine code

Byte code: Bytecode is the compiled format for Java programs. Once a Java program has been converted to bytecode, it can be transferred across a network and executed by Java Virtual Machine (JVM). A Bytecode file generally has a .class extension.

b) Write any two methods of file and file input stream class each.

File class Methods:
(Any 2 methods can be considered) (2 M)

1. public String getName()
   Returns the name of the file or directory denoted by this abstract pathname.
2. public String getParent()
   Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
3. public File getParentFile()
   Returns the abstract pathname of this abstract pathname's parent, or null if this pathname does not name a parent directory.
4. public String getPath()
   Converts this abstract pathname into a pathname string.
5. public boolean isAbsolute()
   Tests whether this abstract pathname is absolute. Returns true if this abstract pathname is absolute, false otherwise
6. public String getAbsolutePath()
   Returns the absolute pathname string of this abstract pathname.
7. public boolean canRead()
   Tests whether the application can read the file denoted by this abstract pathname. Returns true if and only if the file specified by this abstract pathname exists and can be read by the application; false otherwise.
8. public boolean canWrite()
   Tests whether the application can modify to the file denoted by this abstract pathname. Returns true if and only if the file system actually contains a file denoted by this abstract pathname and the application is allowed to write to the file; false otherwise.
9. 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
10. 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.
11. 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 non-directory 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.

12. public long lastModified()
    Returns the time that the file denoted by this abstract pathname was last modified. Returns a long value representing the time the file was last modified, measured in milliseconds since the epoch (00:00:00 GMT, January 1, 1970), or 0L if the file does not exist or if an I/O error occurs.
13. public long length()
    Returns the length of the file denoted by this abstract pathname. The return value is unspecified if this pathname denotes a directory.

FileInputStream class methods :
(Any 2 methods can be considered) (2 M)

1) **int available**() Returns an estimate of the number of remaining bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
2) **void** close() Closes this file input stream and releases any system resources associated with the stream.
3) **int** read() Reads a byte of data from this input stream.
4) int read(byte[] b) Reads up to b.length bytes of data from this input stream into an array of bytes.
5) read(byte[] b, int off, int len) Reads up to len bytes of data from this input stream into an array of bytes.

c) '?' what this operator is called? Explain with suitable example.
(Name of the operator – 1M, Explanation and example – 3 M)

1) '?:' is called as conditional operator or ternary operator.
2) It can be used to check condition in one line, instead of writing if…else statement.
3) Its format is (condtion ? true case : false case)
4) example

```
int a,b;
a=10;
b=(a>5? 12 : 20);
```

Here b= 12 as a is 10 and condition is true. If value of a is changed to 15 then b will have value as 20.

d) Define an exception. How it is handled?

(Definition – 1 M, explanation with all keywords – 3 M)
Exception: An exception is an event, which occurs during the execution of a program, that stop the flow of the program's instructions and takes appropriate actions if handled.
Java handles exceptions with 5 keywords:

1) try 2) catch 3) finally 4) Throw 5) throws

1) **try**: This block applies a monitor on the statements written inside it. If there exist any exception, the control is transferred to catch or finally block.
2) **catch**: This block includes the actions to be taken if a particular exception occurs.
3) **finally**: finally block includes the statements which are to be executed in any case, in case the exception is raised or not.
4) **throw**: This keyword is generally used in case of user defined exception, to forcefully raise the exception and take the required action.
5) **throws**: throws keyword can be used along with the method definition to name the list of exceptions which are likely to happen during the execution of that method. In that case , try … catch block is not necessary in the code.

B. Attempt any one of the following: 6

a) Define a class 'employee' with data members empid, name and salary. Accept data for five objects using Array of objects and print it.

(Class declaration-1 M, Accept data- 1 M, Array of object- 1 M, Display data- 1 M)

```
import java.io.*;
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[5];
for(inti=0; i<5; i++)
{
e[i] = new employee();
e[i].getdata();
}
System.out.println(" Employee Details are : ");
for(inti=0; i<5; i++)
e[i].show();
}
}
```

(Note: Any relevant logic can be considered)

b) Explain with example how to achieve multiple inheritance with interface.

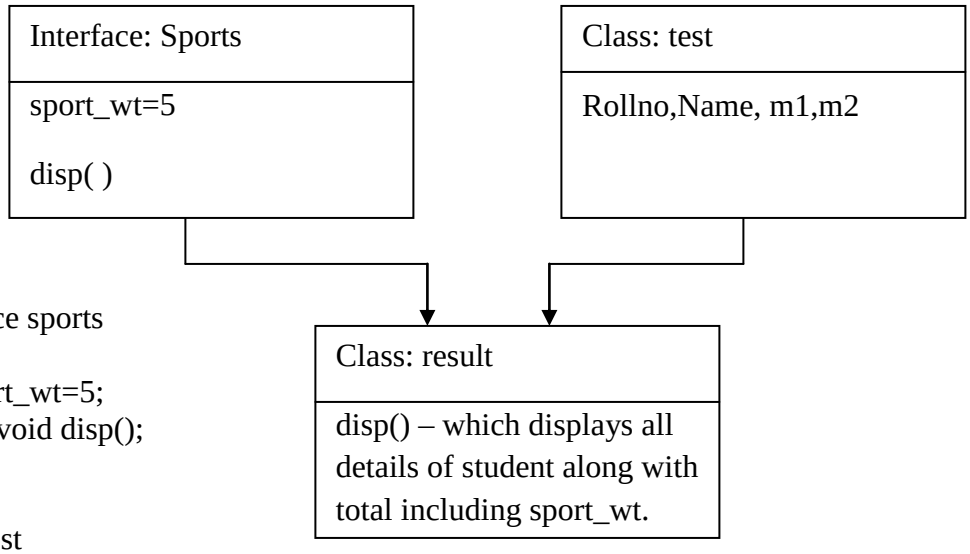
(Multiple inheritance – 2 M, interface – 2 M, example – 2M)

Multiple inheritances: It is a type of inheritance where a derived class may have more than one parent class. It is not possible in case of java as you cannot have two classes at the parent level

Instead there can be one class and one interface at parent level to achieve multiple interface.

Interface is similar to classes but can contain on final variables and abstract method. Interfaces can be implemented to a derived class.

Example :



Code :

```
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();
}
}
```

Q.2. Attempt any Two of the following: 16

a) Define wrapper class. Give the following wrapper class methods with syntax and use:

1) To convert integer number to string.
2) To convert numeric string to integer number.
3) To convert object numbers to primitive numbers using typevalue() method.

(Wrapper class – 2 M, each method syntax and use – 2 M each)

Wrapper Class:

Objects like vector cannot handle primitive data types like int, float, long char and double. Wrapper classes are used to convert primitive data types into object types. Wrapper classes are contained in the java.lang package. The some of the wrapper classes are:

Simple type	Wrapper class
boolean	Boolean
int	Integer
char	Character
float	Float

1) To convert integer number to string : Method is **toString()**
String str = Integer.toString(i) –converts the integer i to string value.

2) To convert numeric string to integer number: Method is **parseInt()**
int i = Integer.parseInt(str) – converts the string value of numeric string str to int i.

3) To convert object number to primitive number using typevalue() method

The method here is typeValue(), meaning that type can be the data type of the number. For example : if x is an Integer object with value 10, then it can be stored as primitive int y with the method intValue() as

```
Integer x = new Integer(10);
int y = x.intValue();
```

Similarly, it can be done with other numeric types as floatValue(), doubleValue() etc.

b) Write syntax and example of

1) **drawstring ()**
2) **drawRect ();**
3) **drawOval ()**
4) **drawArc ()**

(Each method's syntax – 1 Mark, each method's example – 1 Mark)

1) drawstring( )

Displaying String:

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

Syntax:

voiddrawString(String message, int x, int y);

where message is the string to be displayed beginning at x, y

Example:

g.drawString("WELCOME", 10, 10);

2) drawRect( )

Drawing Rectangle:

The drawRect() method display an outlined rectangle. The general forms of this method is :

void drawRect(inttop,intlept,intwidth,int height)

The upper-left corner of the Rectangle is at top and left. The dimension of the Rectangle is specified by width and height.

Example:

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

3) drawOval( )

Drawing Ellipses and circles:

To draw an Ellipses or circles used drawOval() method can be used. The general form of this method is:

Syntax:

voiddrawOval(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);

4) drawArc( )

Drawing Arc:

It is used to draw arc

Syntax:

voiddrawArc(int x, int y, int w, int h, intstart_angle, intsweep_angle);

wherex, 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);

c) What is package? State any four system packages along with their use? How to add class to a user defined packages?

(package – 2 M, Four packages and their use – 1 M each, how to add class to package – 2 M)

Package: 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.

System packages with their use:*(Any 4 can be considered)*

1. **java.lang** - language support classes. These are classes that java compiler itself uses and therefore they are automatically imported. They include classes for primitive types, strings, math functions, threads and exceptions.
2. **java.util** – language utility classes such as vectors, hash tables, random numbers, date etc.
3. **java.io** – input/output support classes. They provide facilities for the input and output of data
4. **java.awt** – set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.
5. **java.net** – classes for networking. They include classes for communicating with local computers as well as with internet servers.
6. **java.applet** – classes for creating and implementing applets.

To add a class to user defined package :

1) Include a **package** command as the first statement in a Java source file.
2) Any classes declared within that file will belong to the specified package.
3) The **package** statement defines a name space in which classes are stored.
4) Example :

```
package MyPack;
public class Balance
{
        .
        .
        .
}
```

Then class Balance in included inside a user defined package 'MyPack'.

Q.3. Attempt any FOUR: 16

a) Differentiate vector and array with any 4 points.
(4M- for any 4 points of differences)

Array	Vector
Array can accommodate fixed number of elements	Vectors can accommodate unknown number of elements
Arrays can hold primitive data type & objects	Vectors can hold only objects.
All elements of array should be of the same data type. i.e. it can contain only homogeneous elements.	The objects in a vector need not have to be homogeneous.
Syntax : Datatype[] arraname= new datatype[size];	Syntax: Vector objectname= new Vector();
For accessing elements of an array no special methods are available as it is not a class , but derived type.	Vector class provides different methods for accessing and managing Vector elements.

b) Write a program to accept a number as command line argument and print the number is even or odd.
(Any other program with correct program may also be considered)
(Logic 2-M, Syntax 2-marks)

```
public class oe
{
public static void main(String key[])
{
int x=Integer.parseInt(key[0]);
if (x%2 ==0)
{
System.out.println("Even Number");
}
else
{
System.out.println("Odd Number");
 }
}
}
```

c) Write a program to copy contents of one file to another file using character stream class.

(Any other program with correct logic may also be considered)
(Logic 2-M, Syntax 2-M)

```
import java.io.*;
class filetest
{
String n=" ";
int rollno=0;
File f1;
 File f2;
FileReader in=null;
FileWriter out=null;
BufferedReader b=new BufferedReader(new
InputStreamReader(System.in));
 void getdata(String s)
{
String ch="y";
try
{
f1=new File(s);
 out= new FileWriter(f1);
while(ch.compareTo("y")==0)
{
System.out.println("Enter name");
n=b.readLine();
System.out.println("Enter number:");
rollno=Integer.parseInt(b.readLine());
out.write(Integer.toString(rollno));
out.write(" ");
out.write(n);
System.out.println("More records :");
ch=b.readLine();
if (ch.compareTo("y")==0)
 out.write('\n');
}
 out.write('\n');
 out.close();
}
catch (IOException e){          System.out.println(e);}
}
 void wordcount(String s)
{
int r=0,count=0,c=0;
```

```
try
{
f1= new File(s);
 in= new FileReader(f1);
while(r!=-1)
{
r=in.read();
 if(r=='\n')
 c++;
if (r==' ' || r=='\n')
{
count++;
}
System.out.println("No. of words:"+count);
System.out.println("No. of lines:"+c);
in.close();
}
catch(IOException e){System.out.println(e);}
}
void display(String s)
{
 int r =0;
 try
{
f1= new File(s);
in = new FileReader(f1);
while(r!=-1)
{
r=in.read();
 System.out.print((char)r);
}
in.close();
 }
catch(IOException e){  System.out.println(e);  }
}
void filecopy(String s)
 {
 int r =0;
try
{
f1= new File(s);
f2=new File("output.txt");
in = new FileReader(f1);
out= new FileWriter(f2);
while(r!=-1)
```

```
{
r=in.read();
out.write((char)r);
}
System.out.println("File copied to output.txt...");
in.close();
out.close();
}
catch(IOException e){System.out.println(e);}
}
public static void main(String[] args)
{
filetest f= new filetest();
String filename= args[0];
f.getdata(filename);
f.filecopy(filename);
f.display(filename);
f.wordcount(filename);
}
}
```

d) State the use of final keyword w. r. t. a method and the variable with suitable example.

All variable and methods can be overridden by default in subclass. In order to prevent this, the final modifier is used.
Final modifier can be used with variable, method or class.

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.
Eg of declaring final variable: final int size = 100;

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.
Eg of declaring a final method:

```
final void findAverage()
{
//implementation
}
```

e) Describe following states of applet life cycle:

a. Initialization state.
b. Running state.
c. Display state.

(**Initialization State 1M, Running state 1M, Display state 2M)**
Initialization state: 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.

Running state: 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
}
```

Display state: 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
}
```

Q.4.

A. Attempt any THREE: 12

a) Compare string class and StringBuffer class with any four points. (Any four points 1M each)

Sr. No.	String	StringBuffer
1.	String is a major class	StringBuffer is a peer class of String
2.	Length is fixed	Length is flexible
3.	Contents of object cannot be modified	Contents of can be modified
4.	Object can be created by assigning String constants enclosed in double quotes.	Objects can be created by calling constructor of StringBuffer class using ‗new'
5.	Ex:- String s=‖abc‖;	Ex:- StringBuffer s=new StringBuffer (―abc‖);

b) Explain thread priority and method to get and set priority values.

Threads in java are sub programs of main application program and share the same memory space. They are known as light weight threads. A java program requires at least one thread called as main thread. The main thread is actually the main method module which is designed to create and start other threads. 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.
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.
To set the priority thread class provides setPriority ()
Syntax: Thread.setPriority (priority value);

To see the priority value of a thread the method available is getPriority().
Syntax: int Thread.getPriority ();

c) Define a class having one 3 digit number, num as data member, initialize and display reverse of that number.
(Logic 2-M, Syntax 2-M)

```
class reverse
{
public static void main(String args[])
{
int num = 253;
int q,r;
for (int I = 0; i<3; i++)
{
q = num / 10;
r = num % 10;
System.out.println(r);
Num = q;
}
}
}
```

d) Write syntax and function of following methods of Date class:

i. getTime ()
ii. getDate ()

(1 M each for syntax, 1 M each for function)

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.

B. Attempt any one of the following: 6

a) Explain following methods of vector class:
- **i. elementAt ()**
- **ii. addElement ()**
- **iii. removeElement ()**

(Syntax 1m each, Use 1M each)

1) **Object elementAt(int index)**
Returns the component at the specified index.
2) **void addElement(Object obj)**
Adds the specified component to the end of this vector, increasing its size by one.
3) **boolean removeElement(Object obj)**
Removes the first (lowest-indexed) occurrence of the argument from this vector.

b) Explain <PARAM> tag of applet with suitable example.

(Syntax of PARAM tag 1M, Use of tag 1m, Any suitable example 2M)
(Any suitable example may be considered)

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.5. Attempt any TWO of the following: 16

a) Write a program to input name and age of a person and throws an user define exception if entered age is negative.

(Declaration of class with proper members and constructor 4 M, statement for throwing exception and main method 4 M)
[Note: Any other relevant program can be considered]

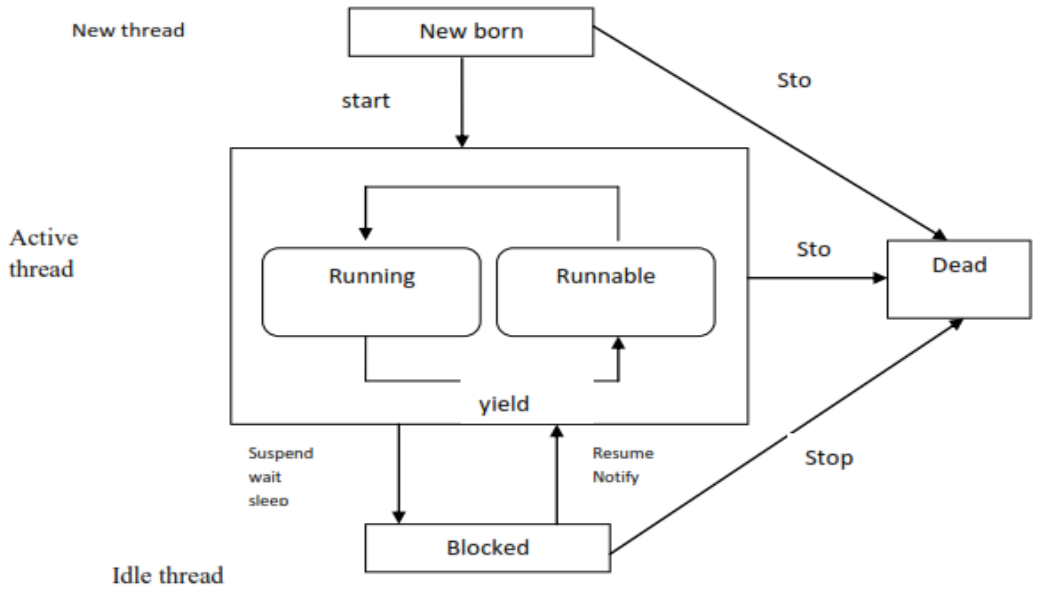
```
import java.io.*;
class negative extends Exception
{
negative(String msg)
{
super(msg);
}
}

class negativedemo
{
public static void main(String ar[])
{
int age=0;
String name;
```

```
InputStreamReader isr=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(isr);
System.out.println("enter age and name of person");
try
{
age=Integer.parseInt(br.readLine());
name=br.readLine();
{
if(age<0)
{
throw new negative("age is negative");
}
else
throw new negative("age is positive");
}}
catch(negative n)
{
System.out.println(n);
}
catch(Exception e)
{
}
}
}
```

b) With suitable diagram explain life cycle of Thread.

(2 M for diagram, 6 M for Explanation)



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 to another by different methods and ways.

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().

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.

suspend() : Thread can be suspended by this method. It can be rescheduled by resume().

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().

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 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. The stop method may be used when the premature death is required.

c) State the use of font class. Describe any three methods of font class with their syntax and example of each.

(Use:- 2M, *Any three method* (syntax and example), *for each method -2M)*
(Any other suitable example may also be considered)

Uses:-

- 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.

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 systemproperty 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 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.6. Attempt any FOUR of the following: 16

a) Differentiate applet and application with any four points.
(Any four points 1 M each)

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

b) Define JDK. List the tools available in JDK explain any one in detail.
(Definition 1 M, list any four tools -2M, explanation of any component -1 M)

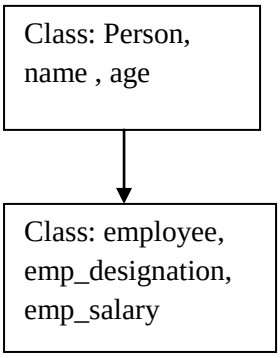
Definition: A Java Development Kit (JDK) is a collection of tools which are used for developing, designing, debugging, executing and running java programs.

Tools of JDK.

java
javap
javah
javadoc
jdb
appletviewer
javac

1. **Java Compiler** – it is used to translate java source code to byte code files that the interpreter can understand.

c) Write a program to implement following inheritance:



(Class and method declaration superclass1-1/2 M, class and method declaration of subclass 1-1/2 M, main method 1M)

(Any other program with correct logic may also be considered)

```
class person
{
String name;
int age;
void accept(String n,int a)
{
name=n;
age=a;
}
void display()
{
System.out.println("name--->"+name);
System.out.println("age--->"+age);
}
}
class employee extends person
{
String emp_designation;
float emp_salary;
void accept_emp(String d,float s)
{
emp_designation=d;
 emp_salary=s;
}
void emp_dis()
{
System.out.println("emp_designation-->"+emp_designation);
System.out.println("emp_salary-->"+emp_salary);
}
}
class single_demo
{
public static void main(String ar[])
{
```

```
employee e=new employee();
e.accept("ramesh",35);
e.display();
e.accept_emp("lecturer",35000.78f);
e.emp_dis();
}
}
```

d) Write the effect of access specifiers public, private and protected in package.
(Each specifier 2 marks)

Visibility restriction must be considered while using packages and inheritance in program visibility restrictions are imposed by various access protection modifiers. packages acts as container for classes and other package and classes acts as container for data and methods.

Data members and methods can be declared with the access protection modifiers such as private, protected and public. Following table shows the access protections

Access modifier → / Access location ↓	public	protected	Friendly (default)	Private protected	private
Same class	yes	yes	yes	yes	yes
Subclass in same package	yes	yes	yes	yes	no
Other classes in same package	yes	yes	yes	no	no
Subclass in other packages	yes	yes	no	yes	no
Non-subclasses in other packages	yes	no	no	no	no

e) What are stream classes? List any two input stream classes from character stream.
(Definition and types- 2 M, any two input classes -2 M)

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