MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

Winter – 19 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.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1.		**Attempt any Five of the following:**	10M
	a	**Define Constructor. List its types.**	2M
	Ans	Constructor: A constructor is a special member which initializes an object immediately upon creation. It has the same name as class name in which it resides and it is syntactically similar to any method. When a constructor is not defined, java executes a default constructor which initializes all numeric members to zero and other types to null or spaces. Once defined, constructor is automatically called immediately after the object is created before new operator completes. **Types of constructors:** 1. Default constructor 2. Parameterized constructor 3. Copy constructor	**Definition:1Mark** **Types: 1 Mark**
	b	**Define Class and Object.**	2M

	Ans	**Class**: A class is a user defined data type which groups data members and its associated functions together. **Objec**t: It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods.	**Definition 1 Mark each**
	c	**List the methods of File Input Stream Class.**	**2M**
	Ans	• void close() • int read() • int read(byte[] b) • read(byte[] b, int off, int len) • int available()	**Any Two Each for 1 Mark**
	d	**Define error. List types of error.**	**2M**
	Ans	• Errors are mistakes that can make a program go wrong. Errors may be logical or may be typing mistakes. An error may produce an incorrect output or may terminate the execution of the program abruptly or even may cause the system to crash. Errors are broadly classified into two categories: 1. Compile time errors 2. Runtime errors	**Definition: 1m List: 1m**
	e	**List any four Java API packages.**	**2M**
	Ans	1.java.lang 2.java.util 3.java.io 4.java.awt 5.java.net 6.ava.applet	**1/2 Marks for one Package**
	f	**Define array. List its types.**	**2M**
	Ans	An array is a homogeneous data type where it can hold only objects of one data type. Types of Array:	**Definition 1 Mark, List 1 Mark**

<table>
<tr><td></td><td></td><td>1)One-Dimensional

2)Two-Dimensional</td><td></td></tr>
<tr><td></td><td>g</td><td>List access specifiers in Java.</td><td>2M</td></tr>
<tr><td></td><td>Ans</td><td>1)public

2)private

3)friendly

4)protected

5)Private Protected</td><td>Any 2, 1M for each</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>2.</td><td></td><td>Attempt any Three of the following:</td><td>12M</td></tr>
<tr><td></td><td>a</td><td>Differentiate between String and String Buffer.</td><td>4M</td></tr>
<tr><td></td><td>Ans</td><td>
<table>
<tr><th>String</th><th>String Buffer c</th></tr>
<tr><td>String is a major class</td><td>String Buffer is a peer class of String</td></tr>
<tr><td>Length is fixed (immutable)</td><td>Length is flexible (mutable)</td></tr>
<tr><td>Contents of object cannot be modified</td><td>Contents of object can be modified</td></tr>
<tr><td>Object can be created by assigning String constants enclosed in double quotes.</td><td>Objects can be created by calling constructor of String Buffer class using “new”</td></tr>
<tr><td>Ex:- String s=”abc”;</td><td>Ex:- StringBuffer s=new StringBuffer (“abc”);</td></tr>
</table>
</td><td>Any 4 Points 4 Marks</td></tr>
<tr><td></td><td>b</td><td>Define a class circle having data members pi and radius. Initialize and display values of data members also calculate area of circle and display it.</td><td></td></tr>
<tr><td></td><td>Ans</td><td>class abc

{</td><td>correct Program with correct logic 4 Mark</td></tr>
</table>

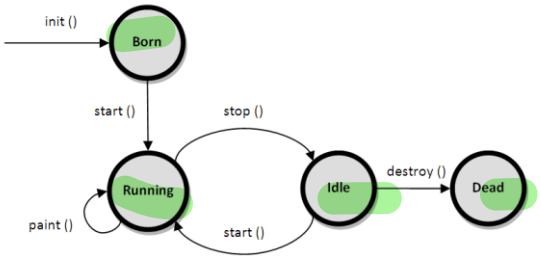
```
float pi,radius;
abc(float p, float r)
{
pi=p;
radius=r;
}
 void area()
{
float ar=pi*radius*radius;
System.out.println("Area="+ar);
}
void display()
 {
System.out.println("Pi="+pi);
System.out.println("Radius="+radius);
} }
 class area
{
public static void main(String args[])
{
abc a=new abc(3.14f,5.0f);
a.display();
```

		a.area();
		}
		}	

	c	**Define exception. State built-in exceptions.**	**4M**
	Ans	An exception is a problem that arises during the execution of a program. Java exception handling is used to handle error conditions in a program systematically by taking the necessary action **Built-in exceptions:** • **Arithmetic exception:** Arithmetic error such as division by zero. • **ArrayIndexOutOfBounds Exception:** Array index is out of bound • **ClassNotFoundException** • **FileNotFoundException:** Caused by an attempt to access a nonexistent file. • **IO Exception:** Caused by general I/O failures, such as inability to read from a file. • **NullPointerException:** Caused by referencing a null object. • **NumberFormatException:** Caused when a conversion between strings and number fails. • **StringIndexOutOfBoundsException:** Caused when a program attempts to access a nonexistent character position in a string. • **OutOfMemoryException:** Caused when there's not enough memory to allocate a new object. • **SecurityException:** Caused when an applet tries to perform an action not allowed by the browser's security setting. • **StackOverflowException:** Caused when the system runs out of stack space.	**Definition 2 Marks, List: 2 Marks**
	d	**Write syntax and example of :**	**4M**

		1) drawRect() **2)drawOval()**	
	Ans	**1)drawRect() :** drawRect () method display an outlined rectangle. **Syntax: void drawRect(int top,int left, int width,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);** **2) drawOval( ):** Drawing Ellipses and circles: 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);**	**drawRect: 2Marks, drawOval: 2 Marks**
3.		**Attempt any Three of the following:**	
	a	**Explain the following classes.** **i)Byte stream class** **ii)Character Stream Class**	**4M**
	Ans	**i)Byte stream class:** 1) **InputStream** and **OutputStream** are designed for byte streams 2) Use the byte stream classes when working with bytes or other binary objects. 3) Input Stream is an abstract class that defines Java's model of streaming byte input	**2M for any two points**

		4)The Input stream class defines methods for performing input function such as reading bytes, closing streams, Marking position in stream. 5) Output Stream is an abstract class that defines streaming byte output. 6) The output stream class defines methods for performing output function such as writing bytes, closing streams **ii)Character Stream Class:** 1. Reader and Writer are designed for character streams. 2. Use character stream classes when working with characters or strings. 3. Writer stream classes are designed to write characters. 4. Reader stream classes are designed to read characters. 5The two subclasses used for handling characters in file are FileReader (for reading characters) and FileWriter (for writing characters).	
	b	**Explain life cycle of Applet.**	**4M**
	Ans	When an applet begins, the AWT calls the following methods, in this sequence: 1. init( ) 2. start( ) 3. paint( ) When an applet is terminated, the following sequence of method calls takes place: 4. stop( ) 5. destroy( )	**1M for diagram ,3M for explanation**



init ( ): The **init( )** method is the first method to be called. This is where you should initialize Variables. This method is called only once during the run time of your applet.

start( ): The **start( )** method is called after **init( ).**It is also called to restart an applet after it has Been stopped. Whereas **init( )** is called once—the first time an applet is loaded—**start( )**is called each time an applet's HTML document is displayed onscreen.

Paint ( ): The **paint ( )** method is called each time your applet's output must be redrawn. Paint ( ) is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, paint( ) is called. The paint ( ) method has one parameter of type Graphics.

Stop ( ): When stop ( ) is called, the applet is probably running. You should use stop ( ) to suspend threads that don't need to run when the applet is not visible.

destroy( ): The destroy ( ) method is called when the environment determines that your applet needs to be removed completely from memory.

	c	**Differentiate between class and interfaces.**	**4M**
	Ans		**1M for each point**

Class	Interface
1)doesn't Supports multiple inheritance	1) Supports multiple inheritance
2)"extend " keyword is used to inherit	2)"implements " keyword is used to inherit
3) class contain method body	3) interface contains abstract method(method without body)

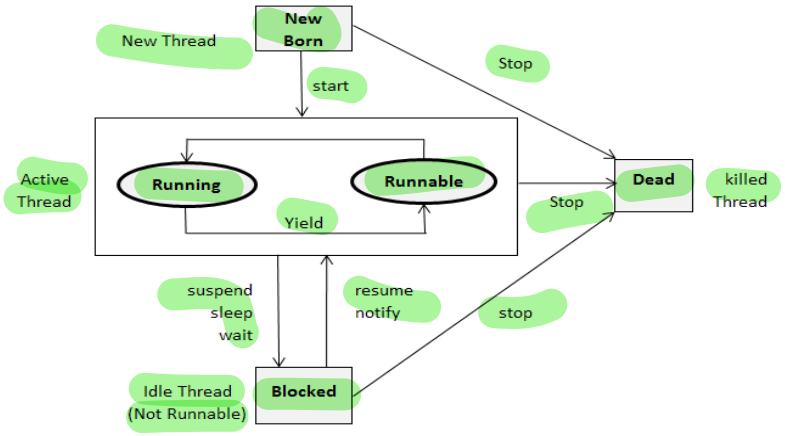
		4)contains any type of variable
		5)can have constructor
		6)can have main() method
		7)syntax
		Class classname
		{
		Variable declaration,
		Method declaration
		}
			4)contains only final variable
			5)cannot have constructor
			6)cannot have main() method
			7)syntax
			Inteface Innterfacename
			{
			Final Variable declaration,
			abstract Method declaration
			}	

	d	**Define type casting. Explain its types with syntax and example.**		**4M**

	Ans	**1.** The process of converting one data type to another is called casting or type casting.
		2. If the two types are compatible, then java will perform the conversion automatically.
		3. It is possible to assign an int value to long variable.
		4. However, if the two types of variables are not compatible, the type conversions are not implicitly allowed, hence the need for type casting.
		There are two types of conversion:
		1.Implicit type-casting:
		2.Explicit type-casting:
		1. Implicit type-casting:
		Implicit type-casting performed by the *compiler automatically*; if there will be no loss of precision.
		Example:
		int i = 3;
		double f;
		f = i;
				1M for definition,3M for types explanation

		output:
		f = 3.0
		
		Widening Conversion:
		
		The rule is to promote the smaller type to bigger type to prevent loss of precision, known as **Widening Conversion**.
		
		2. Explicit type-casting:
		
		• Explicit type-casting performed via a type-casting operator in the prefix form of (*new-type*) operand.
		• Type-casting forces an explicit conversion of type of a value. Type casting is an operation which takes one operand, operates on it and returns an equivalent value in the specified type.
		
		Syntax:
		
		newValue = (typecast)value;
		
		Example:
		
		double f = 3.5;
		
		int i;
		i = (int)f; // it cast double value 3.5 to int 3.
		
		Narrowing Casting: Explicit type cast is requires to Narrowing conversion to inform the compiler that you are aware of the possible loss of precision.	

4.		**Attempt any Three of the following:**	
	a	**Explain life cycle of thread.**	**4M**

	Ans	 Thread Life Cycle Thread has five different states throughout its life. 1. Newborn State 2. Runnable State 3. Running State 4. Blocked State 5. Dead State Thread should be in any one state of above and it can be move from one state to another by different methods and ways. **Newborn state**: When a thread object is created it is said to be in a new born state. When the thread is in a new born state it is not scheduled running from this state it can be scheduled for running by start() or killed by stop(). If put in a queue it moves to runnable state. **Runnable State**: It means that thread is ready for execution and is waiting for 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().	**2M for diagram,2M for explanation**

		Running State: It means that the processor has given its time to the thread for execution. The thread runs until it relinquishes control on its own or it is pre-empted by a higher priority thread. **Blocked state**: A thread can be temporarily suspended or blocked from entering into the runnable and running state by using either of the following thread method. 1) **suspend()** : Thread can be suspended by this method. It can be rescheduled by resume(). 2) **wait()**: If a thread requires to wait until some event occurs, it can be done using wait method and can be scheduled to run again by notify(). 3) **sleep():** We can put a thread to sleep for a specified time period using sleep(time) where time is in ms. It re-enters the runnable state as soon as period has elapsed /over **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.	
	b	**Describe final variable and final method.**	**4M**

	Ans	**Final method**: making a method final ensures that the functionality defined in this method will never be altered in any way, ie a final method cannot be overridden.
		
		Syntax:
		
		final void findAverage()
		
		{
		
		//implementation
		
		}
		
		Example of declaring a final method:
		
		class A
		
		{
			2M for definition,2M for example

```
final void show()
{
System.out.println("in show of A");
}
}
class B extends A
{
void show() // can not override because it is declared with final
{
System.out.println("in show of B");
}}
```

Final variable: the value of a final variable cannot be changed. Final variable behaves like class variables and they do not take any space on individual objects of the class.

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

	c	**Explain any two logical operator in java with example.**	**4M**
	Ans	**Logical Operators:** Logical operators are used when we want to form compound conditions by combining two or more relations. Java has three logical operators as shown in table:	**2M for each operator with eg.**

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

Program demonstrating logical Operators

```
public class Test
```

```
{
public static void main(String args[])
{
boolean a = true;
boolean b = false;
System.out.println("a && b = " + (a&&b));
System.out.println("a || b = " + (a||b) );
System.out.println("!(a && b) = " + !(a && b));
}
}
```

Output:

a && b = false

a || b = true

!(a && b) = true

d **Differentiate between array and vector.** **4M**

Ans **any four points 1m for each point**

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

		2) An array is a homogeneous data type where it can hold only objects of one data type.	2) Vectors are heterogeneous. You can have objects of different data types inside a Vector.
		3) After creation, an array is a fixed-length structure.	3) The size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.
		4) Array can store primitive type data element.	4) Vector are store non-primitive type data element.
		5)Declaration of an array : int arr[] = new int [10];	5)Declaration of Vector: Vector list = new Vector(3);
		6) Array is the static memory allocation.	6) Vector is the dynamic memory allocation.
	e	**List any four methods of string class and state the use of each.**	**4M**
	Ans	The java.lang.String class provides a lot of methods to work on string. By the help of these methods, We can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc. **1) to Lowercase ():** Converts all of the characters in this String to lower case. Syntax: s1.toLowerCase() Example: String s="Sachin"; System.out.println(s.toLowerCase()); Output: sachin **2)to Uppercase():**Converts all of the characters in this String to upper case	**any four methods of string class can be considered**

		Syntax: s1.toUpperCase() Example: String s="Sachin"; System.out.println(s.toUpperCase()); Output: SACHIN **3) trim ():** Returns a copy of the string, with leading and trailing whitespace omitted. Syntax: s1.trim() Example: String s=" Sachin "; System.out.println(s.trim()); Output:Sachin **4) replace ():**Returns a new string resulting from replacing all occurrences of old Char in this string with new Char. Syntax: s1.replace('x','y') Example: String s1="Java is a programming language. Java is a platform."; String s2=s1.replace("Java","Kava"); //replaces all occurrences of "Java" to "Kava" System.out.println(s2); Output: Kava is a programming language. Kava is a platform.	
5.		**Attempt any Three of the following:**	**12-Total Marks**
	a	**Write a program to create a vector with five elements as (5, 15, 25, 35, 45). Insert new element at 2nd position. Remove 1st and 4th element from vector.**	**6M**

	Ans	import java.util.*;
		class VectorDemo
		{
		public static void main(String[] args)
		{
		Vector v = new Vector();
		v.addElement(new Integer(5));
		v.addElement(new Integer(15));
		v.addElement(new Integer(25));
		v.addElement(new Integer(35));
		v.addElement(new Integer(45));
		System.out.println("Original array elements are ");
		for(int i=0;i<v.size();i++)
		{
		System.out.println(v.elementAt(i));
		}
		v.insertElementAt(new Integer(20),1); // insert new element at 2nd position
		v.removeElementAt(0);
		//remove first element
		v.removeElementAt(3);
		//remove fourth element
		System.out.println("Array elements after insert and remove operation ");
		for(int i=0;i<v.size();i++)
		{
		System.out.println(v.elementAt(i));
		}}}
			(Vector creation with elements – 2 M,
			
			Insert new element – 2M,
			
			Remove elements 2 M,
			
			(Any other logic can be considered)

	b	**Define package. How to create user defined package? Explain with example.**	**6M**
	Ans	Java provides a mechanism for partitioning the class namespace into more manageable parts. This mechanism is the package. The package is both naming and visibility controlled mechanism. Package can be created by including package as the first statement in java source code. Any classes declared within that file will belong to the specified package. Package defines a namespace in	**(Definition of package - 1M,**

which classes are stored.

The syntax for defining a package is:

package *pkg*;

Here, pkg is the name of the package

eg : package

mypack;

Package creation - 2M

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

Example - 3M

Syntax:

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

Syntax:

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

(Note Any other example can be considered)

Example:

```
package package1;
public class Box
{
        int l= 5;
        int b = 7;
        int h = 8;
        public void display()
        {
        System.out.println("Volume is:"+(l*b*h));
        }
}
```

Source file:

```
import package1.Box;
 class volume
{
```

		public static void main(String args[])
		{
		Box b=new Box();
		b.display();
		}
		}	

	c	**Write a program to create two threads one thread will print even no. between 1 to 50 and other will print odd number between 1 to 50.**	**6M**

	Ans	import java.lang.*;
		class Even extends Thread
		{
		public void run()
		{
		try
		{
		for(int i=2;i<=50;i=i+2)
		{
		System.out.println("\t Even thread :"+i);
		sleep(500);
		}
		}
		catch(InterruptedException e)
		{System.out.println("even thread interrupted");
		}
		}
		}
		class Odd extends Thread
		{
		public void run()
		{
		try
		{
		for(int i=1;i<50;i=i+2)
		{
		System.out.println("\t Odd thread :"+i);
		sleep(500);
			Creation of two threads 4M
			
			Creating main to create and start objects of 2 threads: 2M
			
			(Any other logic can be considered)

		}
		}
		catch(InterruptedException e)
		{System.out.println("odd thread interrupted");
		}
		}
		}
		class EvenOdd
		{
		public static void main(String args[])
		{
		new Even().start();
		new Odd().start();
		}
		}	

6.		**Attempt any Three of the following:**	**12 M**
	a	**Explain how to pass parameter to an applet ? Write an applet to accept username in the form of parameter and print "Hello <username>".**	**6M**
	Ans	**Passing Parameters to Applet** • User defined parameters can be supplied to an applet using <PARAM…..> tags. • PARAM tag names a parameter the Java applet needs to run, and provides a value for that parameter. • PARAM tag can be used to allow the page designer to specify different colors, fonts, URLs or other data to be used by the applet. **To set up and handle parameters, two things must be done.** 1. Include appropriate <PARAM..>tags in the HTML document. The Applet tag in HTML document allows passing the arguments using param tag. The syntax of <PARAM…> tag <Applet code="AppletDemo" height=300 width=300> <PARAM NAME = name1 VALUE = value1> </Applet> NAME:attribute name VALUE: value of attribute named by corresponding PARAM NAME.	**(Explanation for parameter passing - 3M,** **Correct Program – 3M**

2. Provide code in the applet to parse these parameters. The Applet access their attributes using the getParameter method.

The syntax is : String getParameter(String name);

Program

```
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>
```

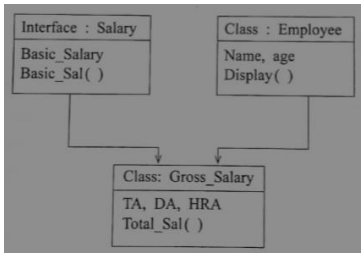
(OR)

```
import java.awt.*;
import java.applet.*;
/*<Applet code = hellouser.class width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</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);
		}
		}	

	b	**Write a program to perform following task** **(i) Create a text file and store data in it.** **(ii) Count number of lines and words in that file.**	**6M**

	Ans	import java.util.*;
		import java.io.*;
		class Model6B
		{
		public static void main(String[] args) throws Exception
		{
		int lineCount=0, wordCount=0;
		String line = "";
		BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
		
		FileWriter fw = new FileWriter("Sample.txt"); //create text file for writing
		System.out.println("Enter data to be inserted in file: ");
		String fileData = br1.readLine();
		fw.write(fileData);
		fw.close();
		BufferedReader br = new BufferedReader(new FileReader("Sample.txt"));
		while ((line = br.readLine()) != null)
		{
		lineCount++; // no of lines count
		String[] words = line.split(" ");
		wordCount = wordCount + words.length;
		// no of words count
		}
		System.out.println("Number of lines is : " + lineCount);
		System.out.println("Number of words is : " + wordCount);
		}
			Create file and store data : 3M,
			
			Get lines and word count : 3M)
			
			(Any other logic can be considered)

		}	
	c	**Implement the following inheritance** 	**6M**

	Ans	interface Salary
		{
		double Basic Salary=10000.0;
		void Basic Sal();
		}
		class Employee
		{
		String Name;
		int age;
		Employee(String n, int b)
		{
		Name=n;
		age=b;
		}
		void Display()
		{
		System.out.println("Name of Employee :"+Name);
		System.out.println("Age of Employee :"+age);
		}
		}
		class Gross_Salary extends Employee implements Salary
		{
		double HRA,TA,DA;
		Gross_Salary(String n, int b, double h,double t,double d)
		{
		super(n,b);
		HRA=h;
			(Interface: 1M,
			Employee class: 2M,
			Gross_Salary class: 3M)

		TA=t;
		DA=d;
		}
		public void Basic_Sal()
		{
		System.out.println("Basic Salary :"+Basic_Salary);
		}
		void Total_Sal()
		{
		Display();
		Basic_Sal();
		double Total_Sal=Basic_Salary + TA + DA + HRA;
		System.out.println("Total Salary :"+Total_Sal);
		}
		}
		class EmpDetails
		{ public static void main(String args[])
		{ Gross_Salary s=new Gross_Salary("Sachin",20,1000,2000,7000);
		s.Total_Sal();
		}
		}
			(Any other logic considered)