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

SUMMER – 19 EXAMINATION

Subject Name: Java Programming **Model Answer** **Subject Code: 17515**

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	**a**	**Attempt any THREE of the following:**	**12 M**
	i	**Define Exception? How is it handled?**	**4 M**
	Ans	**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. .i.e. It is erroneous situation encounter during course of execution of program. Exceptional handling mechanism provides a means to detect errors and throw exceptions, and then to catch exceptions by taking appropriate actions. Java Exception handles as follow • Find the problem (Hit the exception) • Inform that an error has occurred (throw the Exception) • Receive the error information(Catch the exception) • Take corrective action (Handle the Exception) Exception handling in java is done by 5 keywords as: • **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. • **catch:** This block includes the actions to be taken if a particular exception occurs.	Define-1 M , Handling: 3 M

		- **finally:** finally block includes the statements which are to be executed in any case, in case the exception is raised or not. - **throw:** This keyword is generally used in case of user defined exception, to forcefully raise the exception and take the required action. - **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.	
	ii	**WAP to check whether the given number is prime or not.**	**4 M**

	Ans	class PrimeNo
		{
		public static void main(String args[])
		{
		Int num=Integer.parseInt(args[0]);
		int flag=0;
		for(int i=2;i<num;i++)
		{
		if(num%i==0)
		{
		System.out.println(num + " is not a prime number");
		flag=1;
		break;
		}
		}
		if(flag==0)
		System.out.println(num + " is a prime number");
		}
		}
			Correct Program with proper logic 4 M

	iii	**Write syntax to inherit one interface into another interface.**	**4 M**

	Ans	An Interface can extend another interface similarly to the way that a class can extend another class. The extends keyboard is used to extends an interface and the child interface inherits the method of the parent interface.
		Syntax:
		Interface class2 extends class1
		{
		Body of class2
		}
		Example:
			Proper Syntax-4 M

		Interace A
		{
		Int code=11;
		String name=”Computer”;
		}
		Interface B extends A
		{
		Void display();
		}	

	iv	**List and explain Applet attributes**.	**4 M**
	Ans	The HTML APPLET Tag and attributes The APPLET tag is used to start an applet from both an HTML document and from an applet viewer. The syntax for the standard APPLET tag: < APPLET [CODEBASE = codebaseURL] CODE = appletFile [ALT = alternateText] [NAME = appletInstanceName] WIDTH = pixels HEIGHT = pixels [ALIGN = alignment] [VSPACE = pixels] [HSPACE = pixels]> [< PARAM NAME = AttributeNameVALUE = AttributeValue>] [< PARAM NAME = AttributeName2 VALUE = AttributeValue>] . . . </APPLET> • CODEBASE is an optional attribute that specifies the base URL of the applet code or the directory that will be searched for the applet‟s executable class file. • CODE is a required attribute that give the name of the file containing your applet‟s compiled class file which will be run by web browser or appletviewer. • ALT: Alternate Text. The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser cannot run java applets. • NAME is an optional attribute used to specifies a name for the applet instance. • WIDTH AND HEIGHT are required attributes that give the size(in pixels) of the applet display area. • ALIGN is an optional attribute that specifies the alignment of the applet. • The possible value is: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE,	**List 1 Marks and Explain 3 Marks**

		and ABSBOTTOM. • VSPACE AND HSPACE attributes are optional, VSPACE specifies the space, in pixels, about and below the applet. HSPACE VSPACE specifies the space, in pixels, on each side of the applet • PARAM NAME AND VALUE: The PARAM tag allows you to specifies applet- specific arguments in an HTML page applets access there attributes with the getParameter()method.	
	b	**Attempt any ONE of the following:**	**6 M**
	i	**Explain package creation with suitable example.**	**6 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;
		Here, 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|*);
		Example:
		package1:
		package package1;
		public class Box
		{
		int l= 5;
		int b = 7;
		int h = 8;
		public void display()
		{
			Explanation 2 M, Example 4 M

		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();
		}
		}	

	ii	**Explain serialization with suitable example for writing an object into file.**	**6 M**
	Ans	Serialization is the process of writing the state of an object to a byte stream. This is useful when you want to save the state of your program to a persistent storage area, such as a file. At a later time, you may restore these objects by using the process of deserialization. Serialization is also needed to implement Remote Method Invocation (RMI). RMI allows a Java object on one machine to invoke a method of a Java object on a different machine. An object may be supplied as an argument to that remote method. The sending machine serializes the object and transmits it. The receiving machine deserializes it. **Example:** Assume that an object to be serialized has references to other objects, which, in turn, have references to still more objects. This set of objects and the relationships among them form a directed graph. There may also be circular references within this object graph. That is, object X may contain a reference to object Y, and object Y may contain a reference back to object X. Objects may also contain references to themselves. The object serialization and deserialization facilities have been designed to work correctly in these scenarios. If you attempt to serialize an object at the top of an object graph, all of the other referenced objects are recursively located and serialized. Similarly, during the process of deserialization, all of these objects and their references are correctly restored.	Explanation 3 M, Example 3 M
2		**Attempt any TWO of the following:**	**16 M**
	a	**Explain any four methods of graphics class.**	**8 M**
	Ans	**(i) drawOval( )** Drawing Ellipses and circles: To draw an Ellipses or circles used draw Oval() method can be used.	Any four method with

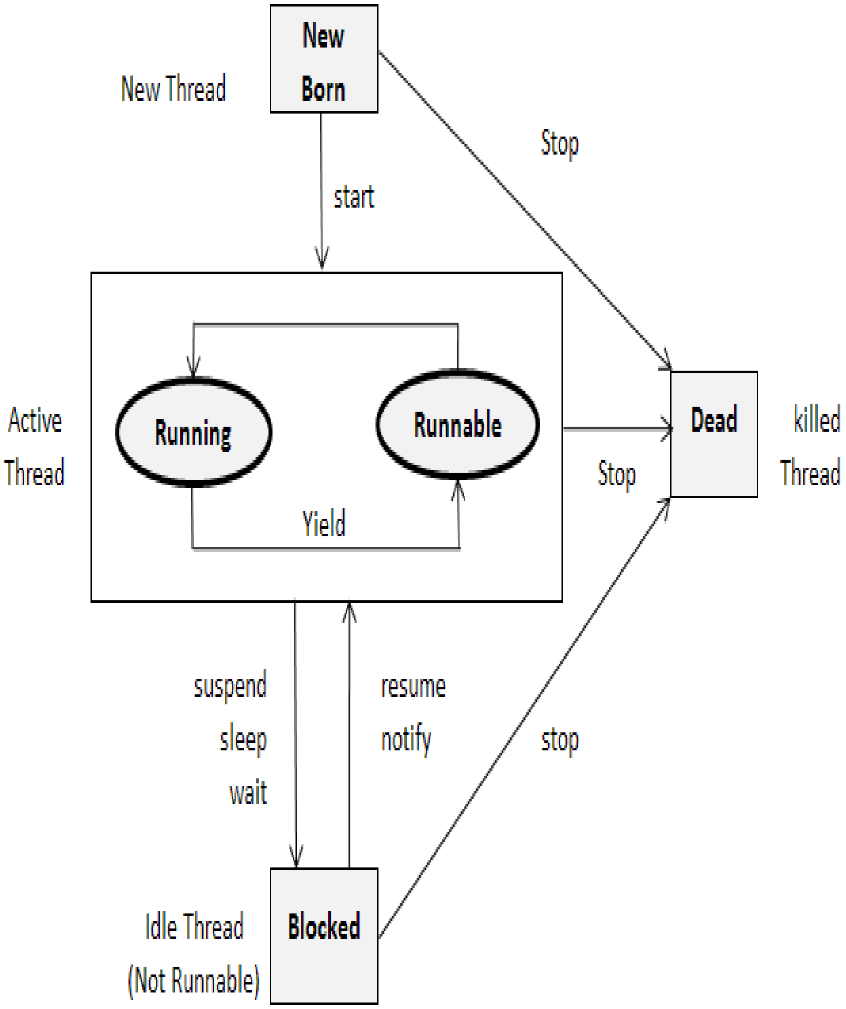
		Syntax: void drawOval(int top, int left, int width, int height);
		The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by top and left and whose width and height are specified by width and height to draw a circle or filled circle, specify the same width and height the following program draws several ellipses and circle.
		Example: g.drawOval(10,10,50,50);
		(ii) drawPolygon
		drawPolygon() method is used to draw arbitrarily shaped figures.
		Syntax: void drawPolygon(int x[], int y[], int numPoints);
		The polygon‟s end points are specified by the co-ordinates pairs contained within the x and y arrays.
		The number of points defines by x and y is specified by numPoints.
		Example:
		int xpoints[]={30,200,30,200,30};
		int ypoints[]={30,30,200,200,30};
		int num=5;
		g.drawPolygon(xpoints,ypoints,num);
		(iii)drawArc( )
		It is used to draw arc.
		Syntax: void drawArc(int x, int y, int w, int h, int start_angle, int sweep_angle);
		where x, y starting point, w & h are width and height of arc, and start_angle is starting angle of arc sweep_angle is degree around the arc
		Example: g.drawArc(10, 10, 30, 40, 40, 90);
		(iv) drawRect()
		The 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);
		
		(v) drawString( )
		Displaying String: drawString() method is used to display the string in an applet window
		Syntax: void drawString(String message, int x, int y);
		where message is the string to be displayed beginning at x, y
		Example: g.drawString("WELCOME", 10, 10);
		(vi) drawLine()
			proper syntax 2 M each

		The drawLine() method is used to draw line which take two pair of coordinates, (x1,y1) and (x2,y2) as arguments and draws a line between them. The graphics object g is passed to paint() method.
		Syntax: g.drawLine(x1,y1,x2,y2);
		Example: g.drawLine(100,100,300,300);	

	b	**WAP to throw authentication failure exception if the user has entered wrong password i.e. accept the password from the user and then rechecked if it is properly entered then valid user exception should throw.**	**8 M**

	Ans	import java.io.*;
		class PasswordException extends Exception
		{
		PasswordException(String msg)
		{
		super(msg);
		}
		}
		class PassCheck
		{
		public static void main(String args[])
		{
		BufferedReader bin=new BufferedReader(new InputStreamReader(System.in));
		try
		{
		System.out.println("Enter Password : ");
		if(bin.readLine().equals("abc"))
		{
		System.out.println("Valid User ");
		}
		else
		{
		throw new PasswordException("Authentication failure");
		}
		}
		catch(PasswordException e)
		{
		System.out.println(e);
		}
		catch(IOException e)
			Proper program with correct logic 8 M

		{
		System.out.println(e);
		}
		}
		}	

	c	**Explain thread life cycle with suitable diagram.**	**8 M**
	Ans	 New Thread New Born Stop start Active Thread Running Runnable Yield Stop Dead killed Thread suspend sleep wait resume notify stop Idle Thread (Not Runnable) Blocked **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.	Diagram 2 M, Explanation: 6 M

		• **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(). • **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. • **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 • **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.	
3		**Attempt any FOUR of the following:**	**16 M**
	a	**Explain this keyword with suitable example.**	**4 M**
	Ans	This keyword: 1. Keyword **'this'** in Java is a reference variable that refers to the current object. 2. It can be used to refer current class instance variable 3. It can be used to invoke or initiate current class constructor 4. It can be passed as an argument in the method call 5. It can be passed as argument in the constructor call 6. It can be used to return the current class instance 7. "this" is a reference to the current object, whose method is being called upon.	Explanation : 2M Any 1 example : 2M Any other example also considered

8. You can use "this" keyword to avoid naming conflicts in the method/constructor of your instance/object.

Example1

Using ‘this’ keyword to refer current class instance variables

```
class Test

{

   int a;

   int b;

   Test(int a, int b)            // Parameterized constructor

   {

      this.a = a;

      this.b = b;

   }

     void display()

   {

      System.out.println("a = " + a + "  b = " + b);

   }

     public static void main(String[] args)

   {

      Test object = new Test(10, 20);

      object.display();

   }

}
```

Example 2: Using ‘this’ keyword to invoke current class method

```
class Test {


   void display()

   {

      // calling fuction show()

      this.show();

         System.out.println("Inside display function");

   }

        void show() {

      System.out.println("Inside show funcion");

   }

   public static void main(String args[]) {

      Test t1 = new Test();

      t1.display();

   }

}
```

	b	**Write a program to add two strings using command line arguments.**	**4 M**

	Ans	class Concats
		{
		public static void main(String args[])
		{
		String s1=args[0]; //Accept first string command line
			Correct Logic : 2M
			Correct syntax : 2M
			(Note: Any other logic can be considered)

```
            String s2=args[1]; //Accept second string command line
            String s3= s1+s2;
            System.out.println("Concatnated String is  : "+s3);
}
}
```

(OR)

```
class Concats
{
public static void main(String args[])
{
            String s1=args[0]; //Accept first string command line
            String s2=args[1]; //Accept second string command line
            String s3=s1.concat(s2);
            System.out.println("Concatnated String is  : "+s3);
}
}
```

c **Give four differences between strings and string buffer class.** **4 M**

Ans

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 object can be modified

Any 4 points : 1M each

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 operator.
5	String s=”MSBTE”	StringBuffer s=new StringBuffer (“MSBTE”)

	d	**Explain thread priority and method to get and set priority values.**	**4 M**
	Ans	**Thread Priority:** In java each thread is assigned a priority which affects the order in which it is scheduled for running. Thread priority is used to decide when to switch from one running thread to another. Threads of same priority are given equal treatment by the java scheduler. Thread priorities can take value from 1-10.	Thread Priority explanation :2M Each Method : 1M

Following are integer constants for Thread Priority:

1) MIN_PRIORITY= The minimum priority of any thread(int value of 1)
2) NORM_PRIORITY= The normal priority of any thread(int value of 5)
3) MAX_PRIORITY= The maximum priority of any thread(int value of 10)

1. **setPriority:**

   Syntax: public void setPriority(int number);

   This method is used to assign new priority to the thread.

2. **getPriority:**

   Syntax: public int getPriority();

It obtain the priority of the thread and returns integer value.

	e	**Write a program to copy the contents of one file into another.**	**4 M**
	Ans		Correct Logic 2M

```
import java.io.*;

class Filecopy

{

  public static void main(String args[]) throws IOException
{
FileInputStream in= new FileInputStream("input.txt"); //FileReader class can be used
FileOutputStream out= new FileOutputStream("output.txt"); //FileWriter class can be used
int c=0;
try
{
  while(c!=-1)
  {
      c=in.read();
      out.write(c);
  }
 System.out.println("File copied to output.txt....");
}
finally
{
if(in!=null)
in.close();
if(out!=null)
out.close();
}
}
}
```

Correct Syntax 2M

	f	**Give four difference between applet and application.**	**4 M**

<table>
<tr><td></td><td><b>Ans</b></td><td>
<table>
<tr><th>Sr.No</th><th>Applet</th><th>Application</th></tr>
<tr><td>1</td><td>Applet does not use main() method for initiating execution of code.</td><td>Application uses main() method for initiating execution of code.</td></tr>
<tr><td>2</td><td>Applet cannot run independently.</td><td>Application can run independently.</td></tr>
<tr><td>3</td><td>Applet cannot read from or write to files in local computer.</td><td>Application can read from or write to files in local computer.</td></tr>
<tr><td>4</td><td>Applet cannot communicate with other servers on network</td><td>Application can communicate with other servers on network.</td></tr>
<tr><td>5</td><td>Applet are restricted from using libraries from other language such as C or C++.</td><td>Application are not restricted from using libraries from other language.</td></tr>
</table>
</td><td>Any 4 points : 1M each</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td><b>4</b></td><td><b>a</b></td><td><b>Attempt any <u>THREE</u> of the following:</b></td><td><b>12 M</b></td></tr>
<tr><td></td><td><b>i</b></td><td><b>Explain shift right and shift left operator.</b></td><td><b>4 M</b></td></tr>
</table>

Ans

The Left Shift (<<): the left shift operator, <<, shifts all of the bits in a 'value' to the left a specified number of times specified by 'num'

General form : value <<num

e.g. x << 2 (x=12)
0000 1100 << 2
= 0011 0000 (decimal 48)

6) The Right Shift (>>): the right shift operator, >>, shifts all of the bits in a 'value' to the right a specified number of times specified by 'num'

General form: value >>num

e.g. x>> 2 (x=32)
0010 0000 >> 2
= 0000 1000 (decimal 8)

Each Bitwise operator explanation: 1 M,

Each example: 1 M

ii List the standard default values of each datatype. **4 M**

Ans

Sr. No	**Datatype**	**Standard default values**
1	byte	0
2	short	0
3	int	0
4	long	0L
5	float	0.0f
6	double	0.0d
7	char	
8	boolean	False

Each ½ mark for correct value/range

iii Explain vector methods: **4 M**

1) **addElement()**

2) **insertElementAt()**

	Ans	**addElement():** It is used to add an object at the end of the Vector.
		Syntax : addElement(Object);
		Example : v.addElement(new Integer(10)); // add Integer object with value 10 at the end of the Vector object 'v'.
		insertElementAt( ) : Adds element to the vector at the location specified by the index.
		Syntax : void insertElementAt(Object element, int index)
		Example: Vector v = new Vector( );
		v.insertElementAt("J", 2); //insert character object in vector at 2nd position.
			Each method syntax : 1M
			and Example 1M

	iv	**Explain stream and various types of streams.**	**4 M**
	Ans	**Stream**: 1. A stream in java is path along which data flows. 2. A stream is an abstraction that either produces or consumes information (i.e. it takes the input or gives the output). 3. A stream presents a uniform, easy-to-use, object oriented interface between program and input/output device. 4. It has source and destination. Java defines two types of streams: based on the datatype on which they operate : 1. **Byte Stream:** Byte streams provide a convenient means for handling input and output of bytes. Byte streams are used, for example, when reading or writing binary data. 2. **Character stream: Character** streams provide a convenient means for handling input and output of characters. They use Unicode and, therefore, can be internationalized. Also, in some cases, character streams are more efficient than byte streams.	Stream explanation : 2M Each type : 1M
	b	**Attempt any <u>ONE</u> of the following:**	**6 M**
	i	**Explain dynamic method dispatch.**	**6 M**

Ans

Dynamic Method Dispatch

1. Dynamic method dispatch is a technique by which call to a overridden method is resolved at runtime, rather than compile time.
2. When an overridden method is called by a reference, then which version of overridden method is to be called is decided at runtime according to the type of object it refers.
3. Dynamic method dispatch is performed by JVM not compiler. Dynamic method dispatch allows java to support overriding of methods and perform runtime polymorphism.
4. It allows subclasses to have common methods and can redefine specific implementation for them. This lets the superclass reference respond differently to same method call depending on which object it is pointing.

Example:

```
class A {

void callme() {

System.out.println("Inside A's callme method");

}

}

class B extends A {

// override callme()

void callme() {

System.out.println("Inside B's callme method");

}

}

class C extends A {

// override callme()
```

Correct Explanation :4M

Example : 2M

(Any other example can be considered)

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

The output from the program is shown here:

```
Inside A’s callme method
Inside B’s callme method
Inside C’s callme method
```

	ii	**Explain class variable with suitable example.**	**6 M**
	Ans	**Class variables** :	Explanation : 3M,

Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.

Example : 3M

(Note: any other example can be considered)

1. There would only be one copy of each class variable per class, regardless of how many objects are created from it.
2. Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.
3. Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
4. Static variables are created when the program starts and destroyed when the program stops.
5. Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
6. Static variables can be accessed by calling with the class name as **ClassName.VariableName.**

Example :

```
public class Employee {
   // salary  variable is a private static variable
   private static double salary;
```

```
     // DEPARTMENT is a constant

     public static final String DEPARTMENT = "Development ";


     public static void main(String args[]) {

        salary = 1000;

        System.out.println(DEPARTMENT + "average salary:" + salary);

     }

   }
```

Output:

```
Development average salary:1000
```

5		**Attempt any TWO of the following:**	**16 M**
	a	**Explain features:** **(i)Platform Independence** **(ii)Robust** **(iii)Dynamic** **(iv)Object oriented**	**8 M**
	Ans	**(i)Platform Independence-** Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs. Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere **(ii)Robust-** Robust simply means strong. Java is robust because: • It uses strong memory management. • There is a lack of pointers that avoids security problems.	2 M for each feature

- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

(iii)Dynamic-

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

(iv)Object oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

	b	**What is inheritance? List types of inheritance and explain single level inheritance with example.**	**8 M**
	Ans	**Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object. The idea behind inheritance is that we can create new classes that are built upon existing classes. When we inherit from an existing class, we can reuse methods and fields of the parent class.	Definition 2M,Types 2M, Example 4M

syntax -

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

A class which is inherited is called a parent or superclass, and the new class is called child or subclass.

Types of inheritance
1)Single inheritance

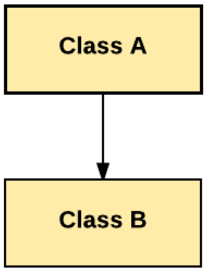
2)Multilevel inheritance

3)Multiple inheritance

4)Hierarchical inheritance

Single level Inheritance

In single inheritance, subclasses inherit the features of one superclass. In image below, the class A serves as a base class for the derived class B.



For example

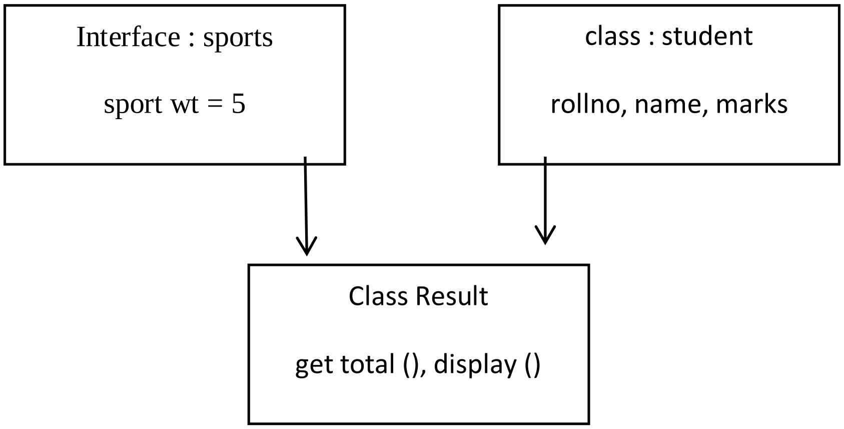
```
 class one
 {
 int v1=10;
void method1()
{
  System.out.println("we are in method 1");
 }
 }
  class two extends one
 {
 void method2()
 {
            System.out.println("v1="+v1);
            System.out.println("we are in method 2");
```

```
                }
        }
         class singleinherit
        {
                public static void main(String[] args)
                {
                        two t = new two();
                     t.method1();
                     t.method2();
                }
        }
```

Output:

```
we are in method 1
v1=10;
we are in method 2
```

c **WAP to implement the following inheritance**: **8 M**



Ans

```
import java.io.*;

 interface sport

{

int  sport_wt=5 ;

}



 class student

{

String name;

int roll_no,marks;



public void getdata()

{

DataInputStream d=new DataInputStream(System.in);

try

{

System.out.println("enter the name");

name=d.readLine();



System.out.println("enter the roll_no");

roll_no=Integer.parseInt(d.readLine());

System.out.println("enter the marks");
```

Correct Logic : 4M

Correct syntax : 4M

(Note: Any other logic can be considered)

```
marks=Integer.parseInt(d.readLine());
}

catch(Exception e)
{
System.out.println(“input output error”);
}
System.out.println(" ");
System.out.println("name="+name);
System.out.println("roll no="+roll_no);
System.out.println("marks1="+marks);
}
}

class result  extends student implements sport
{
int total;
public void get_total()
{
super.getdata();
total=sport_wt+super.marks;
}
```

```
public void display()

{

System.out.println("total marks="+total);

}


public static void main(String args[])

{

result r=new result();

r.get_total();

r.display();

}

}
```

Output:

```
enter the name

abc

enter the roll_no

11

enter the marks

200
```

```
name=abc

roll no=11

marks1=200

total marks=205
```

6		**Attempt any FOUR of the following:**	**16 M**
	a	**WAP to reverse a three digit number accepted from user.**	**4 M**

Ans

```
import java.io.*;

class ReverseNumber

{

  public static void main(String args[])

   {

     int n, reverse = 0;

    DataInputStream d=new DataInputStream(System.in);

try

{

     System.out.println("Enter an integer to reverse");

     n=Integer.parseInt(d.readLine());

     while(n != 0)

     {

         reverse = reverse * 10;

         reverse = reverse + n%10;

         n = n/10;

     }
```

Correct Logic : 4M

Correct syntax : 4M

(Note: Any other logic can be considered

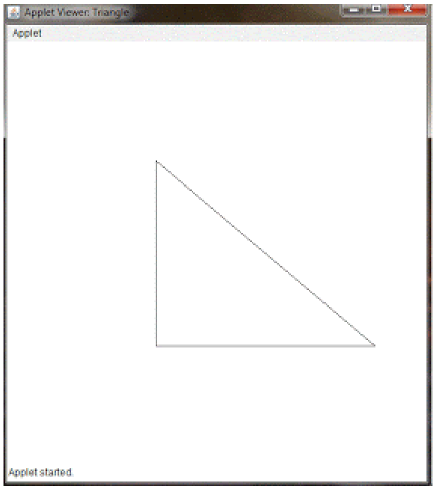
```
          System.out.println("Reverse of the number is " + reverse);


}

catch(Exception e)

{

System.out.println("input output error");

}

}

}

Output:

Enter an integer to reverse

123

Reverse of the number is 321
```

	b	**WAP to draw a triangle inside an applet.**	**4 M**
	Ans		Correct Logic : 2M Correct syntax : 2M (Note: students can use drawPolygon() also)

```
//<applet code=Triangle width=500 height=500></applet>

import java.applet.*;

import java.awt.*;

public class Triangle extends Applet

{

public void paint(Graphics g)

{
```

```
g.drawLine(177,141,177,361);

g.drawLine(177,141,438,361);

g.drawLine(177,361,438,361);

}

}
```

Output



Applet started.

	c	**WAP to display 3 concentric circles in a applet.**	**4 M**

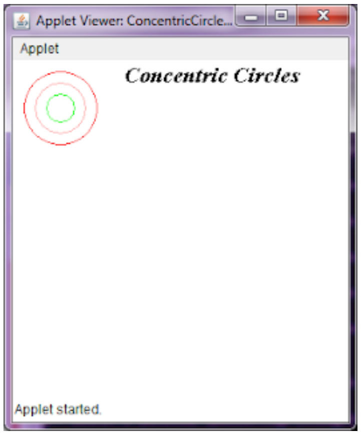
Ans

```
import java.awt.*;

import java.applet.*;



public class ConcentricCircles extends Applet

{

 public void paint(Graphics g)

 {
```

Correct Logic : 2M

Correct syntax : 2M

```
  g.setColor(Color.pink);

  g.drawOval(20,20,45,45);


  g.setColor(Color.red);

  g.drawOval(10,10,65,65);


  g.setColor(Color.green);

  g.drawOval(30,30,25,25);

   }

}

/* <applet code=ConcentricCircles width=300 height=300>

  </applet>

*/
```



	d	**Explain set class with example.**	**4 M**
	Ans	• Set is an interface which extends Collection. It is an unordered collection of objects in which duplicate values cannot be stored. • Basically, Set is implemented by HashSet, LinkedHashSet or TreeSet (sorted representation).	Definition 2M,Example 2M

- Set has various methods to add, remove clear, size, etc to enhance the usage of this interface

Difference between List and Set

A list can contain duplicate elements whereas Set contains unique elements only.

For eg.

```
//  adding elements in Set

import java.util.*;

public class Set_example

{

        public static void main(String[] args)

        {

                // Set deonstration using HashSet

                Set<String> hash_Set = new HashSet<String>();

                hash_Set.add("abc");

                hash_Set.add("xyz");

                hash_Set.add("abc");

                hash_Set.add("java");

                hash_Set.add("subject");

                System.out.print("Set output without the duplicates");

                System.out.println(hash_Set);



        }
}
```

Output:

```
Set output without the duplicates[abc, java, subject, xyz]
```

we have entered a duplicate entity but it is not displayed in the output.

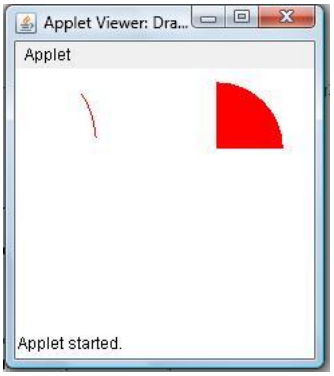
e **Explain draw arc function with example.** **4 M**

Ans The **drawArc( )** designed to draw arcs takes six arguments. The first four are the same as the arguments for drawOval( ) method and the last two represent the starting angle of the arc and the number of degrees around the arc.
The **fillArc( )** method draws a solid arc.

Definition 2M,Example 2M

```
public class DrawArcExample extends Applet
{
        public void paint(Graphics g)
{

setForeground(Color.red);
g.drawArc(10,10,50,100,10,45);

g.fillArc(100,10,100,100,0,90);

}

}
```

Output:



Applet started.

f **Explain two dimensional array in java with example.** **4 M**

Ans An array is a collection of similar type of elements that have a contiguous memory location. **Array** is an object which contains elements of a similar data type.

Definition 1M, Syntax 1M, Example 2M

Types of Array

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Two – dimensional array is the simplest form of a multidimensional array.

Syntax:

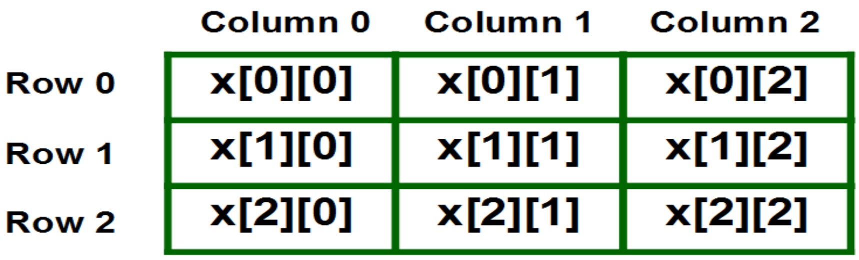
```
data_type[][] array_name = new data_type[x][y];
For example:
int[][] arr = new int[2][3];
```

Initialization

array_name[row_index][column_index] = value;

For example: arr[0][0] = 1;

Representation of 2D array in Tabular Format: A two – dimensional array can be seen as a table with ‘x’ rows and ‘y’ columns where the row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1). A two – dimensional array ‘x’ with 3 rows and 3 columns is shown below:



	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

for eg.

```
class Twodarray
 {
   public static void main(String[] args)
   {
        int[][] arr = { { 1, 2 }, { 3, 4 } };

        for (int i = 0; i < 2; i++)
        {
            for (int j = 0; j < 2; j++)
            {
                System.out.print(arr[i][j] + " ");
            }

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

Output:

```
1 2

3 4
```