Q. 1 A) 1) (1-mark each, any four)

1. Compile & Interpreted: Java is a two staged system. It combines both approaches. First java compiler translates source code into byte code instruction. Byte codes are not machine instructions. In the second stage java interpreter generates machine code that can be directly executed by machine. Thus java is both compile and interpreted language.

2. Platform independent and portable: Java programs are portable i.e. it can be easily moved from one computer system to another. Changes in OS, Processor, system resources won't force any change in java programs. Java compiler generates byte code instructions that can be implemented on any machine as well as the size of primitive data type is machine independent.

3. Object Oriented: Almost everything in java is in the form of object. All program codes and data reside within objects and classes. Similar to other OOP languages java also has basic OOP properties such as encapsulation, polymorphism, data abstraction, inheritance etc. Java comes with an extensive set of classes (default) in packages.

4. Robust & Secure: Java is a robust in the sense that it provides many safeguards to ensure reliable codes. Java incorporates concept of exception handling which captures errors and eliminates any risk of crashing the system. Java system not only verify all memory access but also ensure that no viruses are communicated with an applet. It does not use pointers by which you can gain access to memory locations without proper authorization.

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

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

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

Q.1) 2) (2-marks each)

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 atleast 1 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 affect 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 riority value of a thread the method available is getPriority.

Syntax: int Thread.getPriority ();

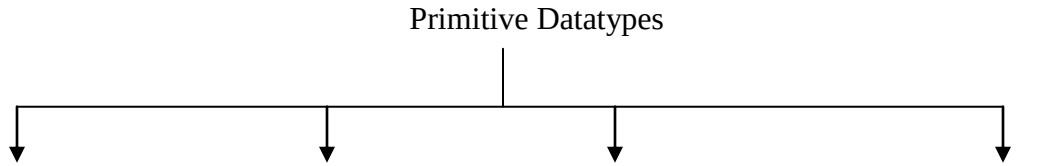
Q.1) 3) [1/2 mark to each data type – 8 x ½=4-marks]

Give all primitive data types available in Java with their storage sizes in bytes.

Java defines eight primitive datatypes as:

byte	float
short	double
long	boolean
char	int

These can be classified into four groups as :



Integer	Floating	Character	Boolean
Byte, short	float	char	Boolean
Int, long	double		

Integer:

All integer datatypes are signed i.e. +ve and –ve. Java does not support unsigned or only +ve integer.

Floating point number:

Also known as real number. It is used for evaluating expression that require fractionsl precesion.

Character:

Java uses Unicode to represent characters. Unicode has a fully international character set for most of the human languages (around 34 thousand character sets from 24 languages). For this purpose it requires 16 bit storage size.

Boolean:

1 – true

0 – false

It can be used for logical values either true or false. It is generally used in 'if' statements for comparison.

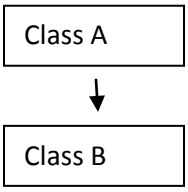
Sr. No	Type	Keyword	Width (bit)
1	long integer	long	64
	short integer	short	16
	integer	int	32
	byte	byte	8
2	double	double	64
	float	float	32
3	character	char	16
4	Boolean	boolean	8

Q.1) 4) [2- marks for explanation, 2-marks for Example]

What is single inheritance? Explain with suitable example.

The mechanism of creating new classes from old one is called as inheritance. The old class is called known as the base class or super class or parent class, & the new one is called as derived or sub or child class.

The inheritance allows subclasses to inherit all the variables of their parent classes. Inheritance can be of different types. Single level inheritance is that in which there is only one super class.



Where Class A is a parent class and class B is derived from A.

Example :

```
//Single level inheritance
import java.lang.*;
class Square       //parent or base class
{
int length;
Square(int x)
        {
        length=x;
        }
        void area()
        {
                int area=length*length;
                System.out.println("Area of Square"+area);
        }
}
class Rectangl extends Square      // child or derived  class
{
        int breadth;
        Rectangl(int x, int y)
        {
                super(x);
                breadth=y;
        }
        void rectarea()
        {
                int area1=length*breadth;
                System.out.println("Area of Rectangle :"+area1);
        }
}

class Shape1
{
        public static void main(String args[])
        {
                Rectangl r = new Rectangl(20,30);
                r.rectarea();
                r.area();
                }
        }
```

```
/* Output
C:\jdk1.3\bin>java Shape1
Area of Rectangle :600
Area of Square400
*/
```

Note : Any other example showing single level inheritance can also be considered.

Q.1 B) [3-marks for explanation, 3-marks for Example]

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 which classes are stored. The syntax for defining a package is:

package package_name;

eg : package mypack;

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 pack1.pack2.pack3; which requires a directory structure as pack1\pack2\pack3.

Example :

package1:

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

source file:

```
import package1.Box;
class volume
{
        public static void main(String args[])
        {
                Box b=new Box();
                b.display();
        }

}

/*OUTPUT:
C:\java programsl>java volume
Volume is:280
*/
```

Note : Any other example showing use of package can also be considered.

Q.1B] 2) ***[3 -marks for Logic, 3-marks for correct syntax]***

```
 //to check whether given number is a palindrome or not.
import java.io.*;
class pallin
{
        public static void main(String args[]) throws IOException
        {
        int num, reversenum = 0, temp;
        BufferedReader br= new BufferedReader( new InputStreamReader(System.in));
        System.out.println("Enter a number to check if it is a palindrome or not\n");
        num= Integer.parseInt(br.readLine());
        temp = num;
                while( temp != 0 )
                {
                 reversenum = reversenum * 10;
                 reversenum = reversenum + temp%10;
                temp = temp/10;
```

```
            }

        if ( num == reversenum )
        System.out.println(num+ " is a palindrome number");
        else
        System.out.println(num+ " is a not a palindrome number");
      }
}
```

Q.2] 1) [4-marks for differences, 2-marks for Array example, 2-marks for Vector example]

Difference :

S.N	Arrays	Vectors
1.	Arrays can accommodate only fixed number of elements.	Vectors can accommodate unknown number of elements.
2.	Arrays can hold primitive data types & objects.	Vectors can hold only objects.
3.	All the elements of the arrays are of same data type. Ie. Homogeneous elements.	The objects in the vectors may not be homogeneous, ie. They can be of different types.
4.	Array is an ordered collection of elements.	Vector is a class contained in java.util package.
5.	Array does not provide any methods to add or remove elements.	Vector class provides elements to add or remove elements.
6.	Syntax : Datatype arrayname = new datatype[size];	Vector objectname = new Vector(); Or Vector objectname=new Vector(size);

Note : Any 4 points can be considered.

Example

```
// Array
class arraytest
{
        public static void main(String args[])
        {
        int arr[]=new int[10];

        //storing 1 to 10 numbers in array
                for(int i=0;i<10;i++)
                {
                arr[i]=i+1;
                }

        //printing elements of array
                System.out.println("Array elements :");
                for(int i=0;i<10;i++)
                {
                System.out.print(arr[i]+"  ");
                }
        }
}
```

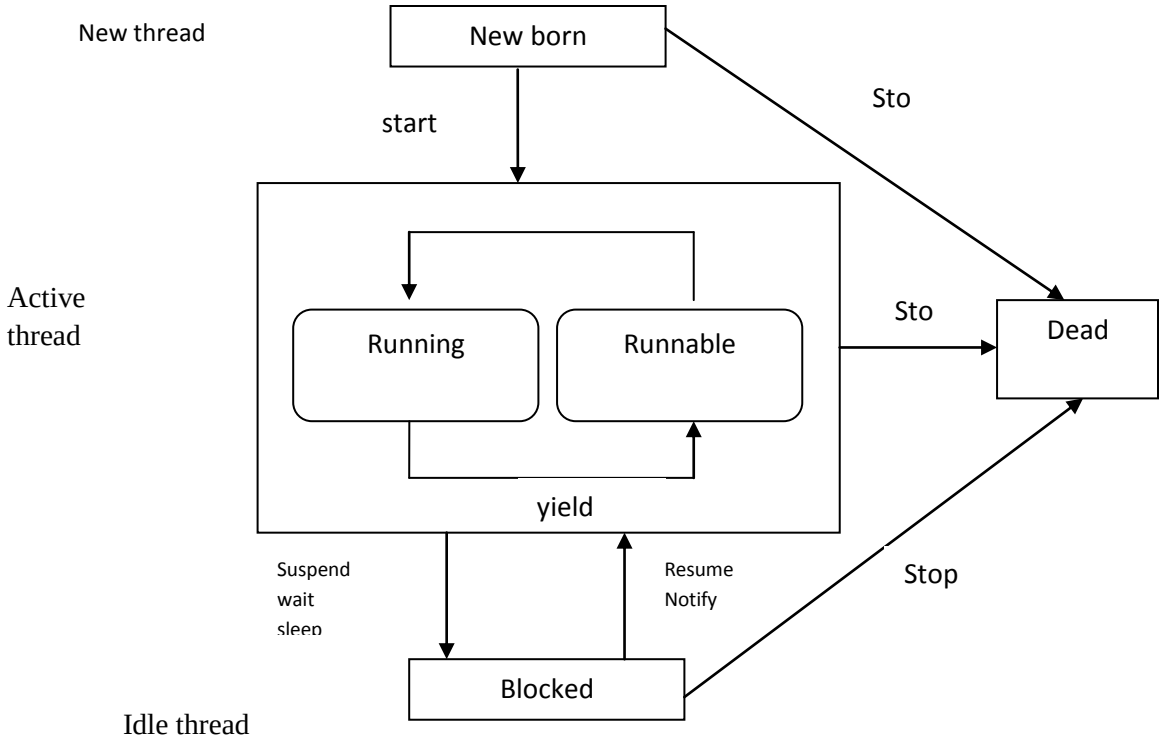
Example 2 :

```
//Vector
import java.util.*;
class vectortest
{
public static void main(String args[])
{
Vector v=new Vector();
Integer p=new Integer(0);
int l=args.length;
for(int i=0;i<5;i++)
{
p=Integer.valueOf(args[i]);
v.addElement(p);
}
int size=v.size();
System.out.println("Elements in vector are:");
```

```
for(int i=0;i<size;i++)
{
System.out.println(v.elementAt(i));
}
}
}
```

Note : Any other examples also can be considered.

Q.2] 2) (2 Marks for diagram , 6 Marks for explanation)



During the lifetime of a thread it passes through different stages / states as:

1. New Born
2. Runnable
3. Running
4. Blocked
5. Dead

A thread is always in one of these 5 states.

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.

Q.2] 3) [4-marks for Logic, 4-marks for correct Syntaxes]

```
import java.lang.Exception;
import java.io.*;
class myException extends Exception
{
        myException(String msg)
        {
        super(msg);
        }
}
class agetest
{
        public static void main(String args[])
        {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                try
                {
                System.out.println("enter the age : ");
                int n=Integer.parseInt(br.readLine());
                        if(n < 18 )
                        throw new myException("Invalid Age");
                        else
                        System.out.println("Valid age");
                }
                catch(myException e)
                {
                System.out.println(e.getMessage());
                }
                catch(IOException ie)
                {}
        }
}
```

Q.3] 1) ***(1 mark each for listing and example 2 marks for description)***

Arithmetic operators are used to construct mathematical expressions

Java provides all the basic arithmetic operators.

The arithmetic operators are:

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

Integer arithmetic:

When both the operands in a single arithmetic expression are integers, the expression is called integer arithmetic.

It always yields an integer value

Real arithmetic:

When the operands of the arithmetic expression are real, the expression is called real arithmetic

Mixed mode arithmetic:

When one of the operands in an expression is real and the other integer, the expression is called mixed mode arithmetic expression.

If either operand is real, then the other is also converted to real and real arithmetic is performed

Example :

```
class ArithmeticOperatorsExample {
  public static void main(String args[]) {
        int a = 10, b = 5, c;
        float d = 4.1f, e = 8.4f,f;
        c = a + b;
        /*Integer arithmetic*/
        System.out.println("The sum of integer numbers a and b is: "+c);
```

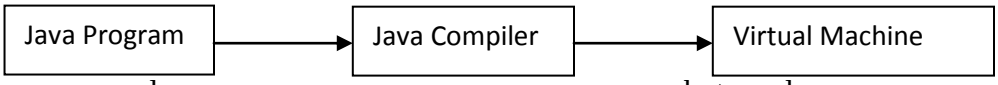
```
        c = a - b;
        System.out.println("The difference of integers a and b is: "+c);
        c = a * b;
        System.out.println("The product of integers a and b is: "+c);
        c = a / b;
        System.out.println("The quotient of integers a and b is: "+c);
        c = a % b;
        System.out.println("The remainder of integers a and b is: "+c);
        /*real arithmetic*/
        f = e + d;
        System.out.println("The sum of real numbers d and e is: "+f);
        f = e - d;
        System.out.println("The difference of real numbers d and e is: "+f);
        f = e * d;
        System.out.println("The product of real numbers d and e is: "+f);
        f = e / d;
        System.out.println("The quotient of real numbers d and e is: "+f);
        f = e % d;
        System.out.println("The remainder of real numbers d and e is: "+f);
        /*mixed mode arithmetic*/
        f = a + d;
        System.out.println("The sum of integer a and real d is: "+f);
        f = a - d;
        System.out.println("The difference of integer a and real d is: "+f);
        f = a * d;
        System.out.println("The product of integer a and real d is: "+f);
        f = a / d;
        System.out.println("The quotient of integer a and real d is: "+f);
        f = a % d;
        System.out.println("The remainder of integer a and real d is: "+f);
    }
}
```

Q.3] 2) (*2 marks for diagram and 2 marks for explanation)*

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



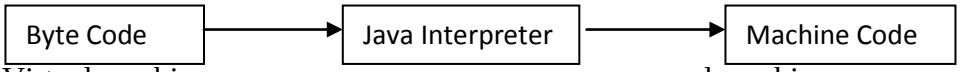
source code byte code

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

Q.3] 3) (Any 4 points, each of 1 mark)

Sr. No	***Class***	***Interface***
1	It can contain abstract methods along with method definitions.	It can contain only method declarations and not method definitions
2	Classes are extended by another class	Interfaces are implemented by classes
3	Methods of objects can be called using the object of the class	Methods of interfaces should be defined in the class which implements the interface
4	The variables of the class can be final/not final	The variables of the interfaces are by default final
5	Object of class can be created	Object of interface cannot be created
6	A class can extend only one class	An class can implement more than one interface
7	Multiple inheritance is not possible with classes	Interface was introduced for the concept of multiple inheritance

8	class Example {
	void method1()
	{
	body
	}
	void method2()
	{
	body
	}
	}
		interface Example{
		int x =5;
		void method1();
		void method2();
		}

Q.3] 4) (Any 4 four, for each listing and explanation 1 mark)

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

Q.3] 5) (logic 3- marks, syntax 1- mark)

```
import java.io.*;
class CopyCharacters  {
        public static void main(String a[]) throws IOException  {
                FileReader fr = null;
                FileWriter fw = null;
                try {
                        fr = new FileReader("sourcefile.txt");
                        fw = new FileWriter("destinationfile.txt");
                        int c;
                        while((c = fr.read())!= -1)  {
                                fw.write(c);
                        }
                }  catch(Exception e) {
                  System.out.println(“Exception caught!!!”);
                }
                finally  {
                        if(fr !=null) {
                                fr.close();
                        }
                        if(fw !=null) {
                                fw.close();
                        }
                }
```

```
        }
    }
```

4. A]1) (Any 4 points, each of 1 mark)

- Applets do not use the main() method for initiating the execution of the code. Applets when loaded automatically call the methods of Applet class to start and execute the applet code whereas application uses the main method for initiating the execution
- Applets cannot run independently. They run from inside a web page using a special feature known as HTML tag whereas application run independently
- Applets cannot read from or write to the files in the local computer whereas the application can
- Applets cannot communicate with other servers on the network whereas applications can
- Applets cannot run any program from local computer whereas application can
- Applets are restricted from using libraries from other languages such as c or c++ whereas applications can use the libraries

Q.4] 2 (Any 2 methods with proper syntax – each for 1 mark. Any two exception each for 1 mark)

boolean canRead() - Tests whether the application can read the file denoted by this abstract pathname

boolean canWrite()

Tests whether the application can modify to the file denoted by this abstract pathname

boolean createNewFile()

Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.

boolean delete()

Deletes the file or directory denoted by this abstract pathname.

boolean equals(Object obj)

Tests this abstract pathname for equality with the given object.

boolean exists()

Tests whether the file or directory denoted by this abstract pathname exists

String getAbsolutePath()

Returns the absolute pathname string of this abstract pathname.

String getName()

Returns the name of the file or directory denoted by this abstract pathname

String getPath()

Converts this abstract pathname into a pathname string.

boolean isDirectory()

Tests whether the file denoted by this abstract pathname is a directory.

Long length()

Returns the length of the file denoted by this abstract pathname.

String toString()

Returns the pathname string of this abstract pathname.

Exceptions:

EOFException – signals that an end of the file or end of stream has been reached unexpectedly during input

FileNotFoundException – informs that a file could not be found

InterruptedIOException – warns that an I/O operation has been interrupted

IOException – signals that an I/O exception of some sort has occurred.

Q.4] 3) ***(logic 3 marks and 1 mark for syntax)***

```
import java.io.*;
class MyException extends Exception {
        MyException(String m) {
                super(m);
        }
}
class ExceptionExample {
        public static void main(String a[]) throws Exception {
                String userName = "Admin";
                String password = "Admin";
                String userInput, userPasswd;
                try {
```

```
                    BufferedReader b = new BufferedReader(new
InputStreamReader(System.in));
                    System.out.println("Enter the user name");
                    userInput = b.readLine();
                    System.out.println("Enter the password");
                    userPasswd = b.readLine();

                    if (userName.equals(userInput)  && password.equals(userPasswd)) {
                        throw new MyException("Welcome!! You are authenticated");
                    } else {
                        throw new MyException("You have entered wrong
information!!");
                    }
                } catch(MyException e)      {
                    System.out.println(e.getMessage());
                }
    }
}
```

Q.4] 4) ***(example 2 marks explanation and syntax 1 mark each)***

for loop is entry controlled loop that provides concise loop control structure

The general structure of for loop is

for (initialization; test condition ; increment)

{

Body of the loop

}

- Initialization of the control variable is done first, using assignment statements e.g.:- int i = 0
- Value of the control variable is tested using the test condition. The test condition is a relational expression, e.g.: i<10, which determine when the loop will exit. If the condition is true the body of the loop is executed else loop is terminated

- After the body of the loop is executed the control transfers back to for statement and the control variable is incremented using statement such as I = i+1

  Example:

```
class ForDemo {
        public static void main(String a[]) {
                System.out.println("Printing the numbers from 1 to 10");
                for(int i = 1; i <=10; i++) {
                        System.out.println(i);
                }
        }
}
```

  The above program is to print numbers from 1 to 10. The loop control variable is set to 1. The number is tested for and printed. After the number is printed, the variable value is incremented by the expression i++. The value of variable is tested and printed. The process is repeated till the value of I reaches 10.

Q.4 B]1 ***(wrapper class definition – 3 marks. a and b sub parts each 1 $\frac{1}{2}$mark)***

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

To convert integer number to string

String str = Integer.toString(i) –converts the integer i to string value

To convert numeric string to integer

```
class IntegerExample
{
```

```
        public static void main(String a[]) {
            String str = Integer.toString(5);
            System.out.println("String value of i "+str );
        }
    }
```

int i = Integer.parseInt(str) – converts the string value of numeric string str to int

```
class StringConversion {
public static void main(String a[]) {
String str = "5";
int i = Integer.parseInt(str);
System.out.println("Integer value of str is "+i);
}
}
```

Q.4 B] 2). ***(explanation for array of objects – 1 mark. 4 marks for logic 1 mark for syntax of program)***

When same type of data is to be stored in contiguous memory allocations arrays are used. When objects are stored in array, it is called array of objects.

```
import java.io.*;
class EmployeeDetails
{
    int empid;
    String name;
    float salary;
    EmployeeDetails() {
        try {
            BufferedReader b = new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("Enter the employee id");
            empid = Integer.parseInt(b.readLine());
            System.out.println("Enter the name");
            name = b.readLine();
```

```
                    System.out.println("Enter the salary");
                    salary = Float.parseFloat(b.readLine());
                }catch(Exception e) {
                }
        }
        public String toString() {
                return "empid: " + empid + " name: " + name + " salary: " + salary;
        }
        public static void main(String a[]) {
                EmployeeDetails[] e = new EmployeeDetails[5];
                for(int i = 0; i<5; i++) {
                        e[i] = new EmployeeDetails();
                }
                System.out.println("The details of five employees are: ");
                for(int i = 0;i<5; i++) {
                        System.out.println(e[i].toString());
                }
        }
}
```

Q 5 1) ***(Explanation of interface – 2 marks, Need of interface – 2 marks Program: 1 mark – declaration of interface, 1 mark - use of interface, 1 mark- Syntax, 1 mark- Logic)***

An interface is a collection of abstract methods and final fields. Interfaces do not specify any code to implement these methods and data fields contain only constants.

An interface is not a class. Writing an interface is similar to writing a class, but they are two different concepts. Therefore, it is the responsibility of the class that implements an interface to define the code for implementation of these methods.

Need of interface: Java does not support multiple inheritance. That is classes in Java cannot have more than one super class. Java allows the use of interface to implement multiple inheritance.

Although a Java class cannot be a subclass of more than one super class, it can implement more than one interfaces.

```
interface Area
{
final static float pi = 3.14F;
float compute(float x, float y);
}

class Rectangle implements Area
{
public float compute(float x, float y)
{
return(x * y);
}
}

class Circle implements Area
{
public float compute(float x,float y)
{
return(pi*x * x);
}
}

class InterfaceArea
{
public static void main(String args[])
{
Rectangle rect = new Rectangle();
Circle cir = new Circle();
Area area;
```

```
area = rect;
System.out.println("Area Of Rectangle = "+ area.compute(5,6));
area = cir;
System.out.println("Area Of Circle = "+ area.compute(10,0));
}
}
```

Q 5 2) (1-mark for each method, 2 marks Syntax, 2 marks- Logic)

Sr. No	Method	Syntax
1.	drawLine()	**public abstract void drawLine(int x1,int y1,int x2,int y2)** Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.
2.	drawRect()	**public void drawRect(int x, int y, int width, int height)** Draws the outline of the specified rectangle. The left and right edges of the rectangle are at x and x + width. The top and bottom edges are at y andy + height. The rectangle is drawn using the graphics context's current color.
3.	drawString()	**public abstract void drawString(String str, int x, int y)** Draws the text given by the specified string, using this graphics context's current font and color. The baseline of the leftmost character is at position

		(x, y) in this graphics context's coordinate system.
4.	drawPolygon()	**public abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)** Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x, y) coordinates defines a point.

```
import java.awt.*;
import java.applet.*;
/*<applet code = grapgr.java width = 250 height = 250></applet>*/
public class grapgr extends Applet
{
public void paint(Graphics g)
{
int xPoints[] = {10,170,80,10};
int yPoints[] = {20,40,140,20};
int nPoints= xPoints.length;
g.drawString("program  using graphics class methods",150,150);
g.drawLine(10,10,50,50);
g.drawRect(10,60,40,30);
g.drawPolygon(xPoints, yPoints, nPoints);
}
}
```

Q5] 3) (Explanation, syntax and use 4 marks 2 marks each for each program)(If PARAM tag is included in the same program, marks should be awarded as Syntax 1 mark, applet tag 1 mark, Logic 2 marks)

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.

```
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);
}
}
```

Q6. 1. (1-mark for each)

a. try : Java uses a keyword try to preface a block of code that is likely to cause an error condition and "throw" an exception. The try block can have one or more statements that could generate an exception. If any one statement generates an exception, the remaining statements in the block are skipped and execution jumps to the catch block that is placed next to the try block.

b. catch: A catch block defined by the keyword catch "catches" the exception "thrown" by the try block and handles it appropriately. The catch block is added immediately after the try block. The catch block can have one or more statements that are necessary to process the exception. Remember that every try statement should be followed by at least one catch statement; otherwise compilation error will occur.

Note that the catch statement works like a method definition. The catch statement is passed a single parameter, which is reference to the exception object thrown (by the try block). If the catch parameter matches with the type of exception object, then the exception is caught and statements in the catch block will be executed. Otherwise, the exception is not caught and the default exception handler will cause the execution to terminate.

c. finally: Java supports another statement known as finally statement that can be used to handle an exception that is not caught by any of the previous catch statements, finally block can be used to handle any exception generated within a try block. It may be added immediately after the try block or after the last catch block.

When a finally block is defined, this is guaranteed to execute, regardless of whether or not in exception is thrown. As a result, we can use it to perform certain house-keeping operations such as closing files and releasing system resources.

d. throw: There may be times when we would like to throw our own exceptions. We can do this by using the keyword throw as follows:

throw new Throwabl'e subclass;

example

throw new ArithmeticException();

throw new NumberFormatException();

Q.6] 2) (2-marks for each)

One use of final keyword is to create named/symbolic constants. Two more uses of final can be seen w.r.t inheritance as:

1. Prevent overriding of method
2. Prevent inheritance

1. Prevent overriding of method:

To disallow a method to be overridden final can be used as a modifier at the start of declaration. Methods written as final cannot be overridden.

e.g.

```
class test
{
final void disp()
        {
                Sop ("In superclass");
        }
}
```

2. Final can be used to even disallow the inheritance, to do this a class can be defined with 'final' modifier, declaring a class as final declares all its method as final

e.g.

```
final class test
{
void disp()
        {
                Sop ("In superclass");
        }
}
Class test 1 extends test // error as class test is final
{
```

…

}

Q.6] 3) ***(2-marks for each)***

An applet developed locally and stored in a local system is known as local applet. When a web page is trying to find a local applet, it does not need to use the Internet connection. It simple searches the directories in the local system and locates and loads the specified applet.

In the case of local applets, CODEBASE may be absent or may specify a local directory.

A remote applet is that which is developed by someone else and stored on a remote computer connected to the Internet. If our system is connected to the Internet, we can download the remote applet onto our system via at the Internet and run it.

In order to locate and load a remote applet, we must know the applet's address on the Web. This address is known as Uniform Resource Locator (URL) and must be specified in the applet's HTML document as the value of the CODEBASE attribute.

e.g. CODEBASE = http://www, .msbte.com/applets

Q.6] 4) Program to find the reverse of a number (2-marks for Logic, 2-marks for Syntax)

```
class Reverse1
{
public static void main(String args[]){
int num = Integer.parseInt(args[0]); //take argument as command line
int remainder, result=0;
while(num>0)
{
remainder = num%10;
result = result * 10 + remainder;
num = num/10;
```

```
}
System.out.println("Reverse number is : "+result);
}
}
```

Q.6] 5) Program to accept two numbers as command line arguments and print the addition of two numbers(2-marks for Logic, 2-marks for Syntax)

```
class commandaddition
{
public static void main(String args[])
{
int num1 = Integer.parseInt(args[0]); //take argument as command line
int num2 = Integer.parseInt(args[0]); //take argument as command line
int Addition = num1 + num2;
System.out.println("Addition of number is : "+Addition);
}
}
```