Q.1) Explain features of java

Java Features:

- The inventors of Java wanted to design a language that could offer solutions to some of the problems encountered in modern programming. They wanted the language not only reliable, portable, and distributed but also simple, compact and interactive. Sun Microsystems officially describes Java with the following attributes:

1. Compiled and Interpreted:

- Generally, in a programming language the source code is first compiled and converted into machine code and then it will be executed. But in Java this thing is different. Java compile first converts the source code into byte code. This byte code is not a machine code. This byte code then Interpreted by java runtime system called Java Virtual Machine (JVM).

2. Platform-Independent and Portable:

- One of the most advantages of Java is its portability. Java program can be easily run on different computer at anytime and anywhere. Java program remains same for any processor, operating systems.
- We can download Java applets from remote computer and run it in over computer. This task is done by Java Virtual Machine which makes it easy to execute.

3. Object Oriented:

- Almost everything in Java is an Object. All program code and data resides within objects and classes. All the object-oriented features like encapsulation, inheritance, and polymorphism are provided in Java. This object-oriented feature make Java program easily understandable and readable.

4. Robust and Secure:

- Java programs are very reliable on any computer system. That is it gives the same output on any system. To better understand how java is robust, consider two of the main reasons for program failure: memory management mistakes and mishandled exceptional condition. Memory management can be a difficult, tedious task in traditional programming environment.
- For example, in C/C++, the programmer must manually allocate and free the dynamic memory. This sometimes leads to problems, because programmer will either forget to free memory that has been previously allocated or, bad tries to free some memory that another part of their code is still using. Java virtually eliminates these problems by managing memory allocation and de-allocation for you using garbage collection

inbuilt facility.

- Security becomes an important issue for a language that is used for programming on Internet. Threat of viruses and abuse of resources is everywhere. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

5. Distributed:

- Java is designed as a distributed language for creating applications on networks. It has the ability to share the data and programs. Java applications can open and access remote objects on Internet as easily as they can do in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project.

6. Multithreaded and Interactive:

- Multithreaded means handling multiple tasks simultaneously. For example, while listening song, one can edit document or scroll the document. Java provides multithreading facility. This means that we need not wait for the application to finish one task before beginning another. This feature greatly improves the interactive performance of graphical applications.

7. High performance:

- Java programs are compiled one time and run one or more time. It takes time only at compilation. After compilation the JVM easily execute this byte code and gives the high performance. Furthermore, multithreading also enhances the performance of execution of java program.

8. Dynamic and Extensible:

- Java is a dynamic language. Java is capable of dynamically linking in new class libraries, methods and objects. Because of this it reduces the memory consumptions when the programs are executed.

Java also supports functions written in other languages such as C and C++.

These functions are known as native methods. Native methods are linked dynamically at runtime

Q.2) Explain different types of array in java

<u>Arrays:</u>

- Array is a collection of values of same type, which are referred by same name. Each value is referred by a number, which is called index number. Arrays of any type can be created and may have one or more dimensions. Each subscript starts with 0.

1. One Dimension Array

- One dimension array has one index number through which an element of the array can be identified. One dimension array can be declare as following:

```
int a[] = new int[5];
```

- This statement will create an array of five elements. If you want to store value 10 into 4^{th} element then you can access 4^{th} element with 3 subscript numbers.

```
a[3]=10;
```

- You can also use these array elements with its subscript number in any mathematical expression.

2. Multidimensional Array

- An array which has more than one dimension then it is called multi-dimension array. The two dimension array is used to store values of a table. First dimension indicates the row index while second dimension indicates column index. The declaration syntax is as given below:

```
Datatype arr[][]=new datatype[totalrows][totalcolumns];
```

- Here this type of array contains (totalrows * totalcolumns) elements. Now lets take an example:

```
int a[][]=new int[3][2];
```

- This array contains 3 rows and 2 columns. Any element can be identified by two subscript number. You can also create three-dimension array. For example, lets declare three-dimension array.

```
int a[][][]=new int[5][5][5];
```

3. Jagged Array

Jagged Array is a special type of multi-dimensional array in which each row has different number of columns. Therefore it is known as variable length array

To declare a Jagged Array, you have to specify only row size and leave the columns size blank.

Following is the syntax to declare a Jagged Array.

Syntax

```
Data_type arr_name[ ][ ]=new data_type[Rowsize][ ];
```

For example :

```
Double d[ ][ ]=new double[5][ ];
```

Int J[ ][ ]=new int[4][ ];

Here, J is a jagged array having 4 rows. Now you can declare the size of columns for each row as shown below :

J[0]=new int[3];
J[1]=new int[2];
J[2]=new int[4];
J[3]=new int[3];

J[0][0]	J[0][1]	J[0][2]	
J[1][0]	J[1][1]		
J[2][0]	J[2][1]	J[2][2]	J[2][3]
J[3][0]	J[3][1]	J[3][2]	

First row i.e.J[0] contains 3 columns, second row (J[1]) has 2 columns, third row (J[2]) has 4 columns and so on.

Following is an example of Jagged Array in which we havae initialized all the rows of array.

Jagged Array Example :

//simple program of jagged array…..

```
class jaggedarryex
{
	public static void main(String args[])
	{
		Int a[ ][ ]={
		{1,2,3,4},
		{1.2),
		{1,2,3},
		{1,2,3,4,5}
		};
	System.out.println("\n Jagged Array…….\n\n");
	for(int i=0;i<a.length; i++)
	{
		for (int j=0; j<a.length; j++)
		{
			System.out.print(a[i][j]+" ");
		}
			System.out.print("\n");
	}
    }
}
```

Q.3 What is overloading? Explain method overloading with example

Overloading methods:

In java it is possible to define two or more methods within the same class that share the same name and parameter declaration is different. When this is the case, the method is said to be overloaded, and the process is referred to as method overloading.

When an overloaded method is invoked, java uses the type and number of parameters as its guide to determine which version of the overloaded method to actually call. Thus, overloaded methods may have differed in the type and number of their parameters.

When java encounters a call to an overloaded method, it simply executes the version of a method whose parameters match the arguments used in the call.

Here is a simple example that illustrates method overloading.

```
class  OverLoad
{
        void test()
        {  System.out.println("No parameter");  }
        void test( int a)
        {  System.out.println("A =  " + a);  }
        void test( int a , int b)
        {  System.out.println("A and B is : " + a + "  " + b );  }
}
class OverLoadDemo
{
        public static void main(String s[])
        {
                OverLoad obj = new OverLoad();
                // call all version one by one
                obj.test();
                obj.test(10);
                obj.test(10,20);
        }
}
```

This program generates following Output.

No parameter
A = 10
A and B is : 10 20

In this program test() method is overload three times. The first version is without parameter, the second takes one parameter and thirds takes two parameters.

As like method you also overload the constructor of the any class to initialize object. Constructor overloading is work as like as method overloading.

In java, constructor overloading is also define as like C++ language.

Q.4 Explain math class with example

- **Math:** The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

Constants	**Desctiption**
static double E	The double value that is closer than any other to *e*, the base of the natural logarithms.
static double PI	The double value that is closer than any other to *pi.*

Method	**Description**
static double abs(double a)	Returns the absolute value of a double value.
static float abs(float a)	Returns the absolute value of a float value.
static int abs(int a)	Returns the absolute value of an int value.
static long abs(long a)	Returns the absolute value of a long value.
static double acos(double a)	Returns the arc cosine of an angle, in the range of 0.0 through *pi*.
static double asin(double a)	Returns the arc sine of an angle, in the range of -*pi*/2 through *pi*/2.
static double atan(double a)	Returns the arc tangent of an angle, in the range of -*pi*/2 through *pi*/2.
static double atan2(double y, double x)	Converts rectangular coordinates (x, y) to polar (r, *theta*).

static double ceil(double a)	Returns the smallest (closest to negative infinity) double value that is not less than the argument and is equal to a mathematical integer.
static double cos(double a)	Returns the trigonometric cosine of an angle.
static double exp(double a)	Returns Euler's number *e* raised to the power of a double value.
static double floor(double a)	Returns the largest (closest to positive infinity) double value that is not greater than the argument and is equal to a mathematical integer.
static double log(double a)	Returns the natural logarithm (base *e*) of a double value.
static double max(double a, double b)	Returns the greater of two double values.
static float max(float a, float b)	Returns the greater of two float values.
static int max(int a, int b)	Returns the greater of two int values.
static long max(long a, long b)	Returns the greater of two long values.
static double min(double a, double b)	Returns the smaller of two double values.
static float min(float a, float b)	Returns the smaller of two float values.
static int min(int a, int b)	Returns the smaller of two int values.
static long min(long a, long b)	Returns the smaller of two long values.
static double pow(double a, double b)	Returns the value of the first argument raised to the power of the second argument.
static double random()	Returns a double value with a positive sign,

	greater than or equal to 0.0 and less than 1.0.
static long round(double a)	Returns the closest long to the argument.
static int round(float a)	Returns the closest int to the argument.
static double sin(double a)	Returns the trigonometric sine of an angle.
static double sqrt(double a)	Returns the correctly rounded positive square root of a double value.
static double tan(double a)	Returns the trigonometric tangent of an angle.
static double toDegrees(double angrad)	Converts an angle measured in radians to an approximately equivalent angle measured in degrees.
static double toRadians(double angdeg)	Converts an angle measured in degrees to an approximately equivalent angle measured in radians.

Ex.

```
Class MethDemo
{
        public static void main(String s[])
        {
                System.out.println("Value of The PI is :-"+Math.PI);
                System.out.println("Value of The E is :-"+Math.E);
                System.out.println("Absulate of 12.67 is :-"+Math.abs(12.67));
                System.out.println("The ceil of the 35.89 is :-"+Math.ceil(35.89));
                System.out.println("The floor value of 35.89 is :-
"+Math.floor(35.89));
                System.out.println("3 to power of 4 is :-"+Math.pow(3,4));
        }
}
```

Q.5 What is Exception?

EXCEPTION HANDLING

- A java exception is an object that describes an exceptional(error) condition that has occurred in piece of code.

- They could be file not found exception, unable to get connection exception and so on.
- Java exception handling is managed via 5 keywords :
    - **try**

      The java code that you think may produce an exception is placed within a try block for a suitable catch block to handle the error. If no exception occurs the execution proceeds with the finally block else it will look for the matching catch block to handle the error. Again if the matching catch handler is not found execution proceeds with the finally block and the default exception handler throws an exception.
    - **catch**

Exceptions thrown during execution of the try block can be caught and handled in a catch block. On exit from a catch block, normal execution continues and the finally block is executed

- **throw**

  To manually throw an exception use the keyword **throw.** It is used to raise an exception within the program, the statement would be throw new exception.

  Syntax :- throw ThrowableInstance;
- **throws**

  throws clause is used to indicate the exceptions that are not handled by the method. It must specify this behavior so the callers of the method can guard against the exceptions. It is specified in the methods signature.

  Syntax :- type methodname(arglist) throws exception-list

  {

  //body of method

  }

  If there are multiple exceptions, they are separated by a comma.

- **finally**

A finally block is always executed, regardless of the cause of exit from the try block, or whether any catch block was executed. Generally finally block is used for freeing resources, cleaning up, closing connections etc. If the finally clock executes a control transfer statement such as a return or a break statement, then this control statement determines how the execution will proceed regardless of any return or control statement present in the try or catch.

- Syntax:-

```
try
{
        //block of code to monitor for errors
}
catch(ExceptionType1 exOb)
{
        //exception handler for ExceptionType1
}
catch(ExceptionType2 exOb)
{
        //exception handler for ExceptionType2
}
finally
{
        //block of code to be executed before try block ends
}
```