# Java Programming

## Introduction:

Java is an object-oriented programming language developed by Sun Microsystems, and it was released in 1995.

We can develop two types of Java programs:

1. Standalone applications
2. Web applets

Executing a standalone Java program involves two steps:

1. Compile source code into bytecode using **javac** compile.
2. Executing the bytecode program using **java** interpreter.

## Simple Java Program

```
Class sampleone {

public static void main(String args[]) {

System.out.println(ñJava is better than c++.ò);

}}
```

## Java Features:

1. Compile and interpreted:
2. Plateform Independent and Portable
3. Oject Oriented
4. Robust and Secure

5. Distriuted: *Java is designed as a distributed language for creating application on networks. It has ability to share both data and programs.*
6. Simple, Small and Familier: *Java does not use pointers, preprocessor header files,* ***goto*** *statement and many others.*
7. High Performance:
8. Dynamic and extensible:*Java is capable of dynamically linking in new class libraries, methods, and objects.*
9. Ease of Development:
10. Scalabitity and Performance:

## Java Program Structure

1. Documentation Section:
2. Package Statement: //optional
3. Import Statement: //optional
4. Interface Statement: //optional
5. Class Definition: //optional
6. Main Method Class {
   Main Method Definition
   }

## Java Statements

1. Empty Statement
2. Labelled Statement
3. Expression Statement
4. Slection Statement
5. Iteration Statement
6. Jump Statement: *continue, break, return and throw.*

## Java virtual Machine

All language compilers translate source code into machine code for a specific computer. Java compiler also does the same thing. Then, how does Java achieve (□ा□त) architecture neturality? The answer is that the java compiler produces an intermediate code known as bytecode for a machine that does not exist. This Machine is called the Java Virtual Machine and it exists only inside the computer memory

## Operators in Java

1. Arithmetic
2. Relational
3. Logical
4. Assignment
5. Increament and decreament
6. Conditional
7. Bitwise (&, !, ^, <<,)
8. Special (.) **ex-** person.age

## Access Control modifiers

- cope only inside the same package (default)
- Scope is visible to world (public)
- Scope of the package and all subclasses (protected)

- Scope only within the classes only (private)

## Non Access Modifiers

### 1. Final

```
class Phone
{
 final int PRICE_MIN = 999;
 final int PRICE_MAX = 5600;//final variable

 final void display()//final method
 {
  System.out.println("Min Price is" + PRICE_MIN);
  System.out.println("Max Price is" + PRICE_MAX );
 }
}
```

### 2. Static

```
class Programming {
  public static void main(String[] args) {
    display();
  }

  static void display() {
    System.out.println("I love to programming in Java.");
  }
}
```

## Casting a value

| Example | Action |
|---|---|
| x  = (int) 7.5 | 7.5 is converted to integer by truncation |
| z = (int) 21.3/(int) 4.5 | Evaluated as 21/4 and the result would be 5 |
| b = (double) sum/n | Division is done in floating point mode |
| y = (int) (a+b) | The result of a+b is converted to integer |

| | |
|---|---|
| z = (int) a+b | á aô is converted to integer then added to b |
| p = cost ((double)x) | Converts á xô to double before using it as parameter |

## The ? Operator

| Using if statement | Using ? operator |
|---|---|
| if(x<0)<br>flag = 0;<br>else<br>flag = 1; | Flag = (x<0) ? 0:1; |

## String methods

| Method call | Task performed |
|---|---|
| s2 = s1.toLowerCase; | Coverts the string s1 to all lowercase |
| s2 = s1.toUpperCase; | Convers the string s1 to all Uppercase |
| s2 = s1.replace(á xô, á yô); | Replace all appearances of x with y |
| s2 = s1.trim(); | Remove white spaces at beginning and end of the string s1 |
| s1.equals(s2) | Returns á true ô if s1=s2 |
| s1.equalsIgnoreCase(s2) | Returns á true ô if s1=s2, ignoring the case of characters |
| s1.length() | Gives the length of s1 |
| s1.ChartAt(n) | Gives nth character of s1 |
| s1.compareTo(s2) | Return negative if s1<s2, positive if s1>s2, and zero if s1=s2 |
| s1.concat(s2) | Concatenates s1 and s2 |
| s1.substring(n) | Givers substring starting from nth character |
| s1.substring(n, m) | Givers substring starting from nth character up to mth (not including mth) |
| String.Valueof(p) | Creates a string object of parameter |

| | |
|---|---|
| | p (simple type or object) |
| p.toString() | Creates a string representation of the object p |
| s1.indexof(ó ô) | Gives the position of the first occurrence of óôin the string s1 |
| s1.indexof(ó ô, n) | Gives the position of óôthat occurs after nth position in the string s1 |
| String.valueof(Variable) | Converts parameter value to string representation |

## Compound assignment operator

```
//Programs to Show How Assignment and Compound Assignment
Operators Works
public class assignmntop {
 public static void main(String[] args) {

  //Simple assigns
  byte bt = 24;
  System.out.println("bt: " + bt);

  //Increments then assigns
  bt += 10;
  System.out.println("bt: " + bt);

  //Decrements then assigns
  bt -= 2;
  System.out.println("bt: " + bt);

  //Multiplies then assigns
  bt *= 2;
  System.out.println("bt: " + bt);

  //Divides then assigns
  bt /= 2;
  System.out.println("bt: " + bt);

//Programs to Show How Assignment and Compound Assignment
```

```
Operators Works
public class assignmntop {
 public static void main(String[] args) {

  //Simple assigns
  byte bt = 24;
  System.out.println("bt: " + bt);

  //Increments then assigns
  bt += 10;
  System.out.println("bt: " + bt);

  //Decrements then assigns
  bt -= 2;
  System.out.println("bt: " + bt);

  //Multiplies then assigns
  bt *= 2;
  System.out.println("bt: " + bt);

  //Divides then assigns
  bt /= 2;
  System.out.println("bt: " + bt);

  //Modulus then assigns
  bt %= 7;
  System.out.println("bt: " + bt);

  //Binary Left Shift  and assigns
  bt <<= 3;
  System.out.println("bt: " + bt);

  //Binary Right Shift and assigns
  bt >>= 4;
  System.out.println("bt: " + bt);

  //Shift right zero fill and assigns
  bt >>>= 1;
  System.out.println("bt: " + bt);
```

```
  //Binary AND assigns
  bt &= 4;
  System.out.println("bt: " + bt);

  //Binary exclusive OR and assigns
  bt ^= 4;
  System.out.println("bt: " + bt);

  //Binary inclusive OR and assigns
  bt |= 4;
  System.out.println("bt: " + bt);
 }
}
```

## Bitwise operator

```
public class bitwiseop {
 public static void main(String[] args) {
  //Variables Definition and Initialization
  int num1 = 30, num2 = 6, num3 =0;

  //Bitwise AND
  System.out.println("num1 & num2 = " + (num1 & num2));

  //Bitwise OR
  System.out.println("num1 | num2 = " + (num1 | num2) );

  //Bitwise XOR
  System.out.println("num1 ^ num2 = " + (num1 ^ num2) );

  //Binary Complement Operator
  System.out.println("~num1 = " + ~num1 );

  //Binary Left Shift Operator
  num3 = num1 <<  2;
  System.out.println("num1 << 1 = " + num3 );

  //Binary Right Shift Operator
  num3 = num1 >>  2;
```

```
  System.out.println("num1 >> 1  = " + num3 );

  //Shift right zero fill operator
  num3 = num1 >>> 2;
  System.out.println("num1 >>> 1 = " + num3 );

 }
}
```

## Instanceof operator

```
class Company {}

public class Employee extends Company {
    public void check() {
        System.out.println("Success.");
    }

    public static void view(Company c) {
        if (c instanceof Employee) {
            Employee b1 = (Employee) c;
            b1.check();
        }
    }

    public static void main(String[] args) {
        Company c = new Employee();
        Employee.view(c);
    }
}
```

## Labeled loop

```
outer: for(int m=1; m<11; m++){
for(int n=1; n<11; n++){
System.out.print(ñò+m*n);
if(n==m)
continue outer;
```

```
}
}
```

## Method overriding

```
class Room{
float length;
float breadth;
Room(float x, float y){ //constructor1
length = x;
breadth = y;
}
Room(float x){ //constructor2
length = breadth = x;
}
int area(){
return (length*breadth);
}
}
```

## Defining and using Static members

```
class mathOperation{
static float mul(float x, float y){
return x*y;
}
static float division(float x, float y){
return x/y;
}
}
class MathApplication{
public static void main(String args[]){
float a = mathOperation.mul(4.0,5.0);
float b = mathOperation.division(a,2.6);
system.out.println(ñb=ò+b);
}
```

```
}
```

## One dimensional array

```
class NumberSorting{
public static void main(String args[]){
int number[] = {55, 26, 47, 87, 34};
int n = number.length;
System.out.print(ñGiven List : ò);
for(int i=0; i<n; i++){
System.out.print(ñò+number[i]);

//sorting begins
for(int i=0; i<n; i++){
for(int j=(i+1); j<n; j++){
if(number[i]<number[j]){
int temp = number[i];
number[i] = number[j];
number[j] = temp;
}
}
}
//sorting end

System.out.println(ñSorted List : ñ);
for(int i=0; i<n; i++){
System.out.print(ñ ñ+number[i]);
}
}
}
```

## Two dimensional array

```
class mulTable{
final static int Row = 0;
final static int Column = 20;
public static void main(String args[]){
```

```
int product[][] = new int[Row][Column];
int row, column;
System.out.println(ñò);
int i,j;
for(i=0; i<Row; i++){
for(j=0; j<Column; j++){
product[][] = i+j;
System.out.print(ñò+product[i][j]);
}
System.out.println(òò);
}
}
}
```

## Interface

```
package my_package;
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 (pi*x*y);
        }
}

class interface_test{
        public static void main(String args[]) {
        Rectangle rect = new Rectangle();
        Area area;
        area = rect;
        System.out.println("Area of Rectangle is
"+area.compute(10,20));
        }
}
```

## Multithreading

```
class A extends Thread{
	public void run() {
		for(int i=1;i<=5;i++) {
			System.out.println("Class A: "+i);
		}
	}
}

class B extends Thread{
	public void run() {
		for(int j=1;j<=5;j++) {
			System.out.println("Class B: "+j);
		}
	}
}

class C extends Thread{
	public void run() {
		for(int k=1;k<=5;k++) {
			System.out.println("Class C: "+k);
		}
	}
}


public class multithreads {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
new A().start();
new B().start();
new C().start();
	}
}
```

## Exception Handling

```
public class exceptionHandling {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
int a=10;
int b=5;
int c=5;
int x,y;

try {
	x = a/(b-c);
}
catch(ArithmeticException e) {
	System.out.println("Division By Zero");
}
y=a/(b+c);
System.out.println(+y);
	}
}
```

## Applet

```
//Simple Appplet Program
import java.awt.*;
import java.applet.*;
public class hello extends Applet{
	public void paint(Graphics g){
		g.drawString("hello",10,10);

	}
}
```

//in html

```
Hello.html
<html>
```

```
<head>
<title>applet</title>
</head>
<body>
<applet code=hello.class width=400 height=200>
</body>
</html>
```

**Cmd for this**

```
javac hello.java
appletviewer  hello.html
```

## Inheritance

```
//Simple Inhetitance
class Room{
        int length;
        int breadth;
        Room(int x, int y){
                length = x;
                breadth = y;
        }
        int area(){
                return (length*breadth);
                }
}

class BedRoom extends Room{
        int height;
        BedRoom(int x,int y,int z){
                super(x,y);   //pass value to supercclass
                height = z;
        }
        int volume(){
                return (length*breadth*height);
        }
}
```

```
public class test {
public static void main(String args[]){
BedRoom br = new BedRoom(12,34,67);
int area = br.area();       //superclass method
int vol1 = br.volume();     //baseclass method
System.out.println("Area = "+area);
System.out.println("Volume = "+vol1);
}
}
```

## Enhanced for loop

```
public class Enhanced_for_loop {
public static void main(String[] args){
String states[] = {"UtterPardesh", "AndhraPardesh", "Haryana"};

for(String i:states){ //Enhanced for loop
        System.out.println("State name: "+i);
}
}
}
```

## Mathematical functions

```
public class mathematical_functions {
public static void main(String[] args){
        int a=2;
        int b=3;
        System.out.println("sin of a is "+Math.sin(a));
        System.out.println("cos of a is "+Math.cos(a));
        System.out.println("tan of a is "+Math.tan(a));
        System.out.println("asin of a is "+Math.asin(a));
        System.out.println("acos of a is "+Math.acos(a));
        System.out.println("atan of a is "+Math.atan(a));
        System.out.println("atan2 of a and b is "+Math.atan2(a,b));
        System.out.println("pow of a raised to b is "+Math.pow(a,b));
```

```
        System.out.println("exp of a is "+Math.exp(a));
        System.out.println("log of a is "+Math.log(a));
        System.out.println("sqrt of a is "+Math.sqrt(a));
        System.out.println("ceil of a is "+Math.ceil(a));
        System.out.println("floor of a is "+Math.floor(a));
        System.out.println("rint of a is "+Math.rint(a));
        System.out.println("round of a is "+Math.round(a));
        System.out.println("abs of a is "+Math.abs(a));
        System.out.println("max of a and b is "+Math.max(a,b));
        System.out.println("min of a and b is "+Math.min(a,b));
}
}
```

## Ternary Operator

```
public class ternary_operator{
        public static void main(String[] args){
                int a=3;
                int b=1;
                int x = (a<b)?a:b;
                System.out.println("Answere is "+x);
        }
}
```