**Java**

**Introduction0**

- Java was developed by James Goslin in 1995 at Sun-Microsystems

**Introduction**

- A function is a block of code that perfomes a task

- Every java program must have atleast on function and that function is **main()**

- Class is a container for related functions

- Every java program must have atleast one class , which contains main() function , i.e

public class **M**ain(){

public void main(){

}

}

- In java we use different coventions for naming classes and methods . Like ,**P**ascal**N**aming**C**onvention for classes and **c**amel**N**aming**C**onvention/**camelCaseNotation** for naming the methods

- A method is a function that belongs to a class

- In a java prpgram , each classes and methods should have access modifiers .

- Access modifier is a special keyword that determines if other classes and methods are accessible in a particular class or program . Ex:- public , private etc

- We use a package to group related classes , interfaces and enums into a single module :

- By convention the base package for a java project is the domain name of your company in reverse . ex:- **package com.company;**

- **System.out.print("Hello World")**; 🡨In this statement , **System** is a class belongs to **java.lang** package , **.** is an operator , **out** is the object and **println()** is method of the class .

**JDK , JRE and JVM**

**JDK**

- JDK is a cross-platformed software development environment that offers a collection of tools and libraries

necessary for Java-based software applications and applets . JDK = JRE + Dev-tools

- It includes JRE , an interpreter/loader(**java)** , a compiler**(javac**) , an archiver(jar) , documentation generator(Javadoc) and other tools needed in Java development

- JDK is what we need to compile the Java source-code and It is the core package used in Java which contains JRE(Java Runtime Environment) and JVM(Java Virtual Machine)

- Applet is Java program that can be embedded into a web-page . It runs inside the web-browser and works at client-side

**JRE**

- JRE(Java Runtime Environment) is what we need to run a java program and contains a set of libraries and other files that JVM uses at the run time . JRE = JVM + Library Classes

- It is an installation package that provides an environment to only run(not develop) the java program(or application) onto your machine .

**JVM**

- JVM(Java Virtual Machine) - It is an abstract machine offers the runtime environment for codes to be executed .Basically , whatever java program we run using JDK or JRE goes into JVM and JVM is responsible for executing java prpgram line by line . hence it is also known as **interpreter**

\* full description on GFG : https://www.geeksforgeeks.org/differences-jdk-jre-jvm/

|  |  |  |
| --- | --- | --- |
| JDK | JRE | JVM |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**How Java codes executed under the hood**

- Basically there are two steps involved here : Compilation and Execution

- In Compilation step , IDE uses java compiler to compile the souce-code(Main.java) into a different format called Java Byte-Code(Main.class)

- Main.class is the Bytecode-representation of the java file

-In our java-project , **src** directory holds our source-code and , **out** directory holds the results of the compilation . Hence , Java Bytecode(**Main.class)** is also stored in **out** folder

- The Java Byte-code(Main.class) is platform-independent and That means , it can run over any OS that has JRE

- JRE has JVM that takes the Java Bytecode and translate it into the native code for the underlying OS . Due to this architecture is the reason why Java applications are portable or platform independent .

**Data-Types**

**Primitive-types :** for storing simple values . In it , we don’t need to allocate and release memory for the variable as it is done by JRE .

int a = 5 ; 🡨 Here , 5 is stored in the location of variable **a** .

int b = a ; 🡨Here , actual value of a is assigned to **b** and stored at its location in memory.

**a = 7 ;**

System.out.println(b);

**output: 5** (b not updated alongwith a)

- Above a and b are having different memory locations . Hence , a and b are completely independent

of each other . Or , if we change the value of a to 7 , the value at b is not gonna change at all .

|  |  |  |
| --- | --- | --- |
| Type | Bytes | Range |
| **byte** | 1 | [-128,127] |
| **short** | 2 | [-32k,32k] |
| **int** | 4 | [-2B,2B] |
| **long** | 8 |  |
| **float** | 4 |  |
| **double** | 8 |  |
| **char** | 2 | A,B,c,d |
| **boolean** | 1 | true/false |

int viewCount = 123456789 or 123\_456\_789(**<2B**)

**long** viewCount = 3\_123\_456\_789 (**>2B**) , Even though we have specified long data-type , it will still act like an integer by default and hence will give error. Hence , we need to add ‘**L/l**’ suffix after the value , to make it work

**long** viewCount = 3\_123\_456\_789**L**

**float** price = 10.99 , Here , Java sees this number-value as a **double** by default and hence will throw error . Hence ,here we need to add suffix **‘F/f’**

**float** price =10.99**F**

**Reference-types:** for storing complex objects . We need to allocate memory for this variable but release is handled by JRE .

Ex:- Date **now** = **new** Date(); Here , **now** is an instance/object of Date class and it has the access to all the members of Date() class . like , **now.getTime();**

Point point1 = new Point(1,1);

Point point2 = point1;

**point1.x = 2;**

System.out.println(point2);

**Output : java.awt.Point[x=2,y=1]** (point2.x updated alongwith point1.x)

- JRE will allocate some memory for **Point(1,1)** object and point1 label is attached to a separate memory location and in that memory location the address of the Point(1,1) object will be stored instead storing the Point(1,1) object .

- When we declare a primitive variable , the value that we assigned in that variable will be stored in that memory location of the variable but , in Reference-type the variable is going to store/hold the address of that Point(1 , 1) object in memory not the actual Point() object .

- Hence , point1 and point2 are storing the address of Point() object or these are the references to the Point() object in the memory .

- Hence , point1 and point2 are not independent of each-other as they are referencing the same object in the memory .

- Changes made to point1 will be reflected in point2 , as can be seen in the output

**- Hence Reference-Types are copied by the references and Primitive-types are copied by values**

**Strings**