How Java Program Works:

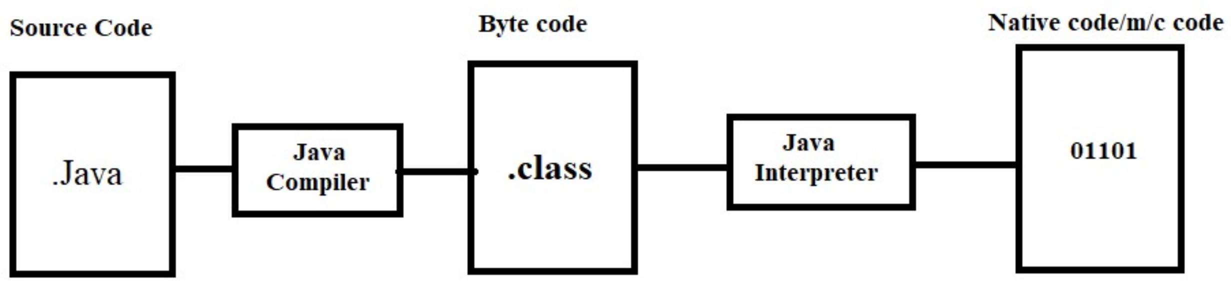


Figure: Program Execution Flow

This diagram will show you in detail description.

1) Java compiler will check syntax error.
2) Java interpreter will convert byte code into native code.
3) After converting into native code or machine code, we can able to see output into console.

Java Compiler:

The process of converting entire source code into byte code that process is known as Java Compiler.

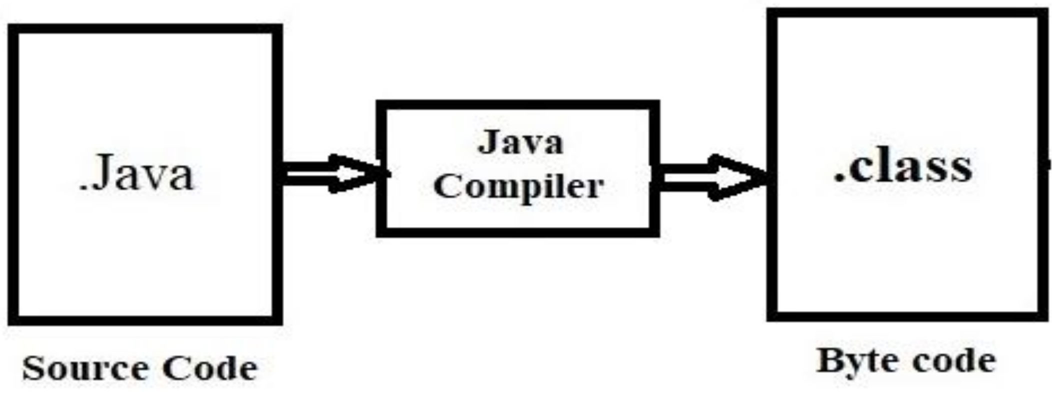


Figure. JAVA Compiler

It is the program that contain set of instruction which is implementer in C/C++ programming Language.

It just check your syntax and grammatical mistakes.

Java Interpreter:

The process of converting byte code into native code, that process is known as Java interpreter.

It is nothing but exe file which is implemented in C or C++ programming language.

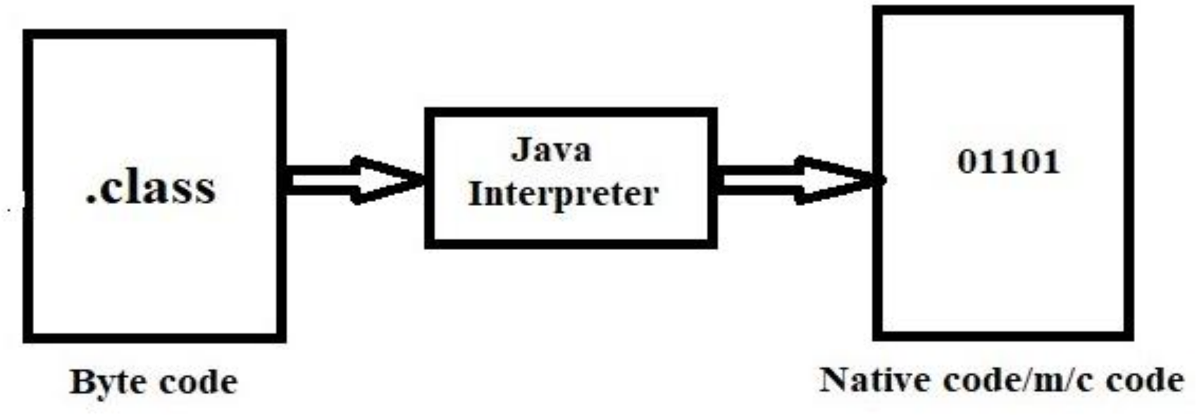


Figure. JAVA Interpreter

What is JDK ?

JDK is **JAVA Development Kit**, JDK contains JRE+ Development tools as JAVA, JAVAC and JAVADOCS.

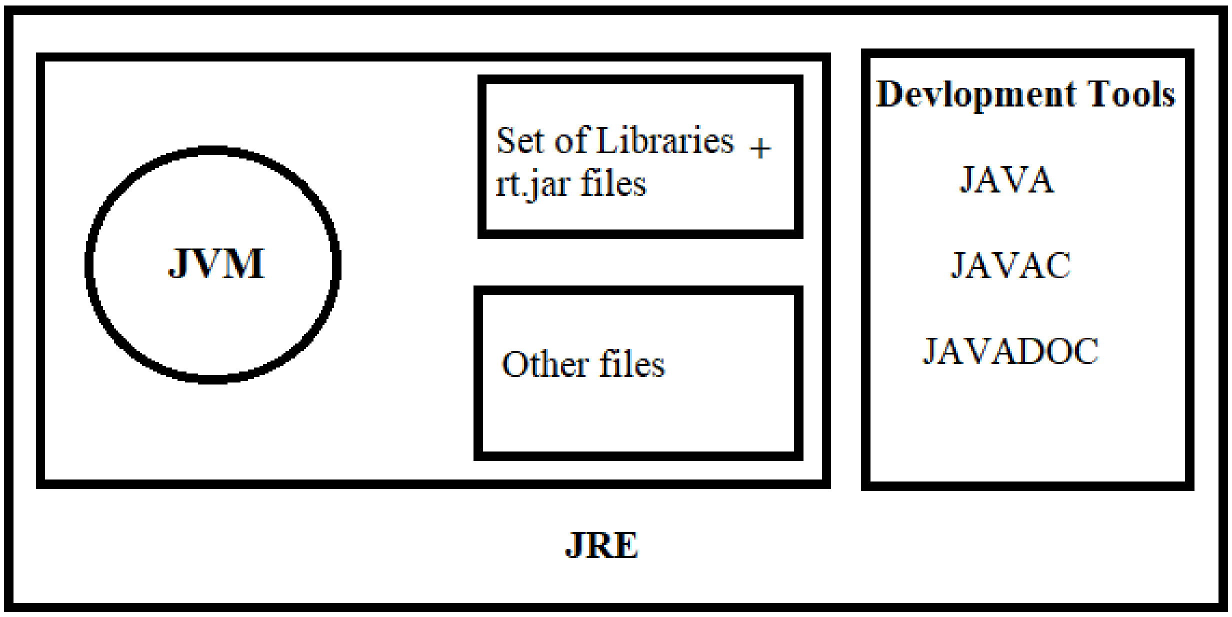


Figure.JDK

What is JRE?

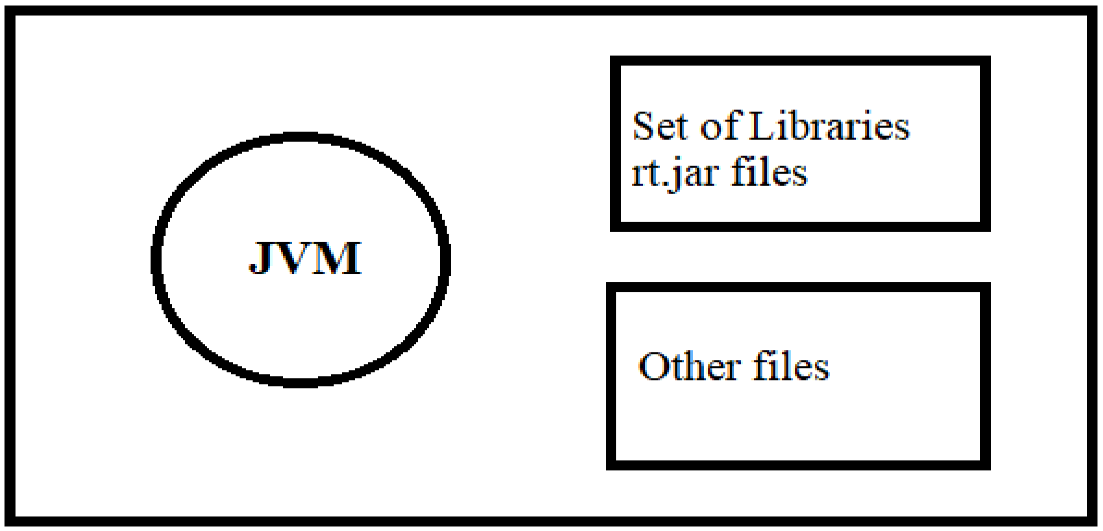


Figure: JRE

JRE stands for **Java Runtime Environment**. It is a set of component to create and run a java application.**JRE** includes **JVM**, set of libraries, other files & rt. jar.

What is JVM?

- **JVM** stands for **Java virtual Machine. JVM** is part of JRE.
- JVM is run time environment in which application is executed. IT is platform dependent.
- JVM is the one that actually calls the main method present in a java code.
- JVM is responsible to Load and Run Application.
- JVM perform following task:
  - ➢ Loads the code.
  - ➢ Verifies code.
  - ➢ Execute code.
  - ➢ Provide runtime environment.
  - ➢ Provide Library.

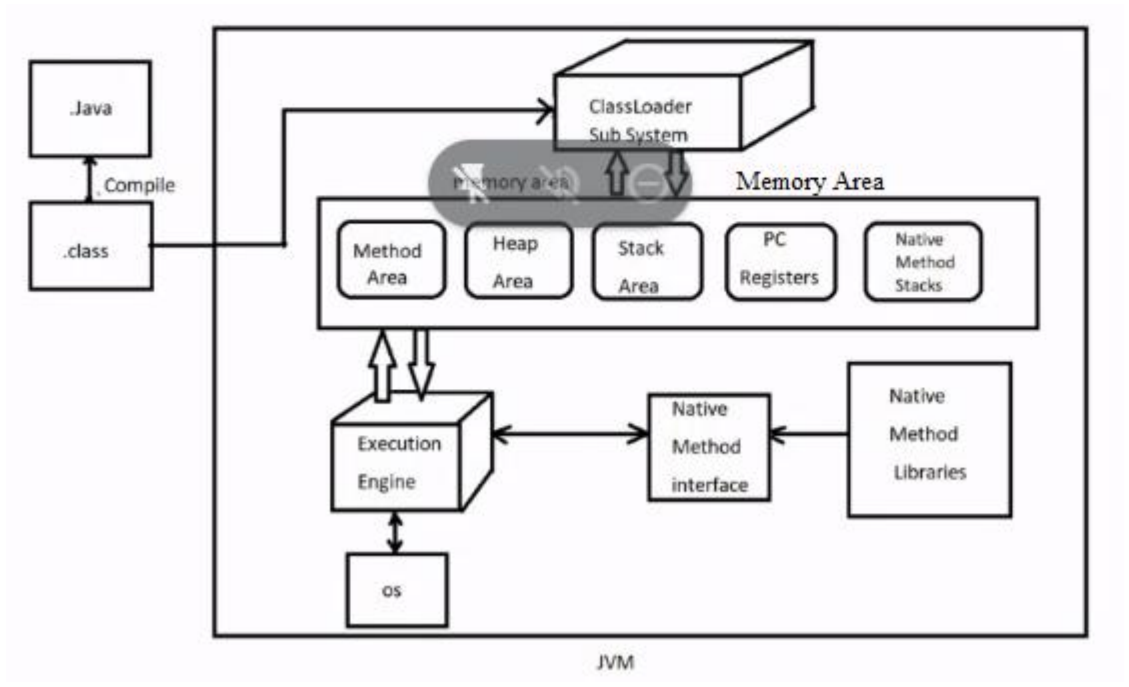


Figure :JVM

When we compile **a.java file, .class** file with same name is generated by JAVA compiler. This **.class** files goes into various step when we run it. These step together describe the whole JVM.

Class Loader:

- Class Loader is responsible for Loading, Linking and Initialization.
- In loading class reader reads the **.class** file, generate the corresponding binary data and save it in method area.
- It stores information like class name, immediate super class name, method, variable, modifier information.
- In linking it perform verification, preparation and resolution.
- Verifies the correctness of file and generated by valid compiler or not.
- In preparation JVM allocates memory for class variable and initialize it.
- in Initialization all static variable are assigned with value assign in code.

Memory Area:

1) **Method area/class area:**
    a) Method area is created when JVM is started.
    b) It stores all class level information like class name, immediate super class name, variable, method name & static variable.
    c) There is only one method area per JVM, and it is in shared resource.

2) Hear Area:

a) Heap area is create when JVM is started.
b) It stores all information of object, instance variable & array.
c) There is also one Heap Area per JVM. It is also a shared resource.

3) Stack Area:

a) It stores the current running method & local variable.
b) When the method is completed, the corresponding entry from the stack will be removed.
c) For every thread JVM will create one stack and it is not shared resource.

4) PC register:

a) It holds the address of next executing instructions.
b) Each thread has separate PC registers.

5) Native method stacks:

a) For every thread JVM will create a separate native method stack.
b) It stores native method information.

Execution Engine:

- This is the central component of JVM.
- Execution engine is responsible to execute java class files.
- Execution engine contains 2 components for executing java classes.

1. **Interpreter:**
    - It is responsible to read byte code and interpret (convert) into machine code (native code) and execute that machine code line by line.
    - The problem with interpreter is it interpreters every time even the same method multiple times. which reduces performance of the system.
    - To overcome this problem sun people introduced JIT compiler in 1.1 version.

2. **JIT Compiler(Just in Time):**
    - The main purpose of JIT compiler is to improve performance.
    - Internally JIT compiler maintains a separate count for every method whenever JVM come across any method call.
    - JVM interprets total program line by line at least once.
    - JIT compilation is applicable only for repeatedly invoked method but not for every method.
3. **Garbage Collector:**
    - It destroy un-referenced objects.

Java Native Interface(JNI):

- It is an interface that interacts with the Native Method Libraries and provides the native libraries(C,C++) required for the execution.
- It enable JVM to call C/C++ libraries and to be called by C/C++ libraries which may be specific to hardware

Native Method Libraries:

- It contains set of libraries which are written in another programming language such as C or C++ for other application.

Features of JAVA

JAVA is having below features:

1. Simple.
2. Platform Independent
3. Neutral
4. Portable
5. Secure.
6. Object Oriented
7. Multi-Threaded.
8. Robust.
9. Distributed.
10. Compile & Interpreted.
11. High Performance.
12. Dynamic
13. Network Savvy.

1) **Simple:**

- Java is very simple & easy to learn.
- We can write java programs very easily no need to write any header files like C and C++.
- To learn Java no any prerequisites required.
- Most of the common complex & confusing features of other language(C, C++) like pointers, multiple inheritance, etc. removed from Java.

2) **Platform Independent:**

- If we write Java Program once, we can run on any platform i.e Java follows Write Once Run Anywhere Principle(WORA).
- We can run same java program on Windows, Linux & MAC also.

3) Architecture Neutral:

- Java programs never communicates with the platform directly. Changes & upgrades in Operating Systems, processors and system resources will not any changes of Java program.
- Suppose we have one program which is already running on machine that uses windows OS and then we upgraded to latest version, RAM size will be change from 4GB to 8GB. No need to modify existing program, it will work on upgraded machine as its not communicating with OS.

4) Portable:

- In JAVA we convert source code to byte code that can run on any platform. That's why JAVA is portable.
  Mobile number portability in India from one operator to another, for that no need to change number similarly byte code(class file) can run on any platform.

5)Secure:

- With the help of java we can develop virus fee, tamper-free system.
- It is very secure language which in most used in banking industry.
- Java program never communicates directly with the machine. First converted into **Byte code** and then converted into **machine code** by the **JVM**.
- If the **Byte code** contains any problem or if it not properly formatted, then **JVM** won't allow that code to run & will raise Verify Error. Internally inside **JVM**, **Byte code** verifier is responsible to verify **Byte code**.
- Hence, Java programs won't cause any problem to the system.
- Some list of Security features are encryption, decryption and cryptography.

6) Object Oriented Programming Language:

- Everything in JAVA in object, without object we can't do anything in JAVA.
- Java is object oriented programming languages like **C++,** most of the times in java, we have to handle everything in terms of object.
- Java provide support for the OOP's features like Inheritance, Polymorphism, Encapsulation, Association, Abstraction, Class & Objects.

7) Multithreaded:

- In the case of multithreading, multiple threads can run simultaneously and can perform specified task simultaneously, so that performance of the application will be improved.
- Java provides inbuilt support for multithreading by providing a rich API.
- It stores the common memory for example we typing some code in notepad and listening the music so these perform simultaneously.

8) Robust:

- Java is strongly typed language Compiler will check every& every declaration & assignment at compile time only compatibility. If any problem w.r.t. types, then at compile time only we can identify the problem.
- There is no need of pointer concept as like C to access the string, array and files etc.
- It provides garbage collector for automatic memory management. Hence, there is no
- Java provides inbuilt Exception Handling. Which prevents abnormal termination of the program at runtime.
- Java is platform independent and it can run on any platform.
- Because of all these facilities, the chance of failing the program at runtime is very less and hence, java is Robust.

9) Distributer:

- If the application is distributed across multiple machine (JVMs) such type of application is called Distributed application**.**
- Java provides inbuilt support for distributed programming with RMI (Remote Method Invocation).

10) Compiled & Interpreted:

- Java is both Compiled & Interpreted Programming language. First java compiler compiles Java code and generates machine independent Byte code.
- At runtime **JVM** interprets this **byte code** into **machine code** by using **Just in Time compiler (JIT)** and executes that **machine code.**

11) High Performance:

- Java is relatively faster than traditional interpreted languages, since the byte code is close to native code.
- It provides high performance due to **JIT** compiler. Because it has more information available.
- **JIT** compiler has more information with it like which code is executed frequently, which class has been loaded and optimize the code for better speed.

12) Dynamic:

- In the case of Java programs, all **.class** files won't be loaded at the beginning. At runtime if **JVM** required any class then only the corresponding **.class** file will be loaded(Dynamic Loading).
- The main advantage is program will always get latest version of **.class** file and memory utilization will be improved.

13) Network Savvy:

- It provides facility to access the JAVA application form access the internet via URL's.
- It also supports the socket programming for sending the message form client to server.

Class:

- Class is blueprint or template from which object is created. Class can access from outside via object.
- A class is a group of member functions(methods), variable, data type, printing statement, scanning object, conditional statement, loop statement, block/body etc.
- A class can be define behavior/state of that object of its support.

Why we write class?

Because we want to design something in the class. Class contains variable and method. We rarely used int and float in class, generally we prefer String in Java for storing values of variable.

Syntax of class

```
<Access Specifier> class  <className> {

// class body here

}
```

There are Two Types of classes

1. Built in classes in Java:
2. User defined classes in Java

1) Built in classes in Java:

Which class bundled within predefined packages in java that are known as Built in classes in Java.

Some example and majority used built in classes in Java are

java.lang.String

java.lang.Object

java.lang.System

java.lang.Exception

java.lang.Class

java.util.Date

java.util.HashMap

java.util.ArrayList

java.util.Iterator

java.util.Thread

2) User defined classes in Java

as name suggests, a custom or user defined class is a class that is created by user. It will contains the class member as defined by user.

Class Consist of:

1) Field:
   Field of class are used to define the properties or state attributes of the class objects. Thus they are declared within the class body. General syntax to declared a class field in given below:
   Example:

```
public class student{
        public int a=10;
         // Where public is access specifier
           int is data type
           a is field or variable name
           10 is value
}
```

2) Method:
   A method in Java is collection of statements which determine the behaviour of a class object.
3) Constructor.
4) Block.
5) Nested classes (Class within the other class is known as nested class).

Summary:

While creating a new class under package we need to follow some steps:

1) Firstly right click on created package then choosing new option then click on class.
2) We get new java class window, in that we need to give class name whatever you want.
3) A Java class must have the class keyword followed by class name.
4) While giving name to the class there are some conditions like-
   - Class name usually starts with Uppercase letter.
   - Do not accept the space between the name.
     If you have lengthy name you can provide" _" in between the name or every first latter of the world should be capital.
     e.g: StudentDetail
   - Do not accept special characters except dollar symbol '$' & Underscore '_'.
   - A Java class can only have public or default access specifier.
5) It can extends only one parent class. By default access extend.
   java.lang.Object directly or indirectly.
6) A class members must be always declared within the set of curly braces{ }.
7) Class containing the main ( ) method is known as main class as it will be act as entry point of your program.

Object:

Object in Java is real world entity which has its own state & behavior.

A JAVA program can have as many objects as required.

An object in JAVA typically contain following thing:

1) State: This is represented by attributes & properties of object.
2) Behaviour: This is defined by method or Object.Iin other words, its functionality.
   e.g. Mobile is real world entity in which states are colour, size, brand, model etc. & behaviour is calling, text, msg etc.

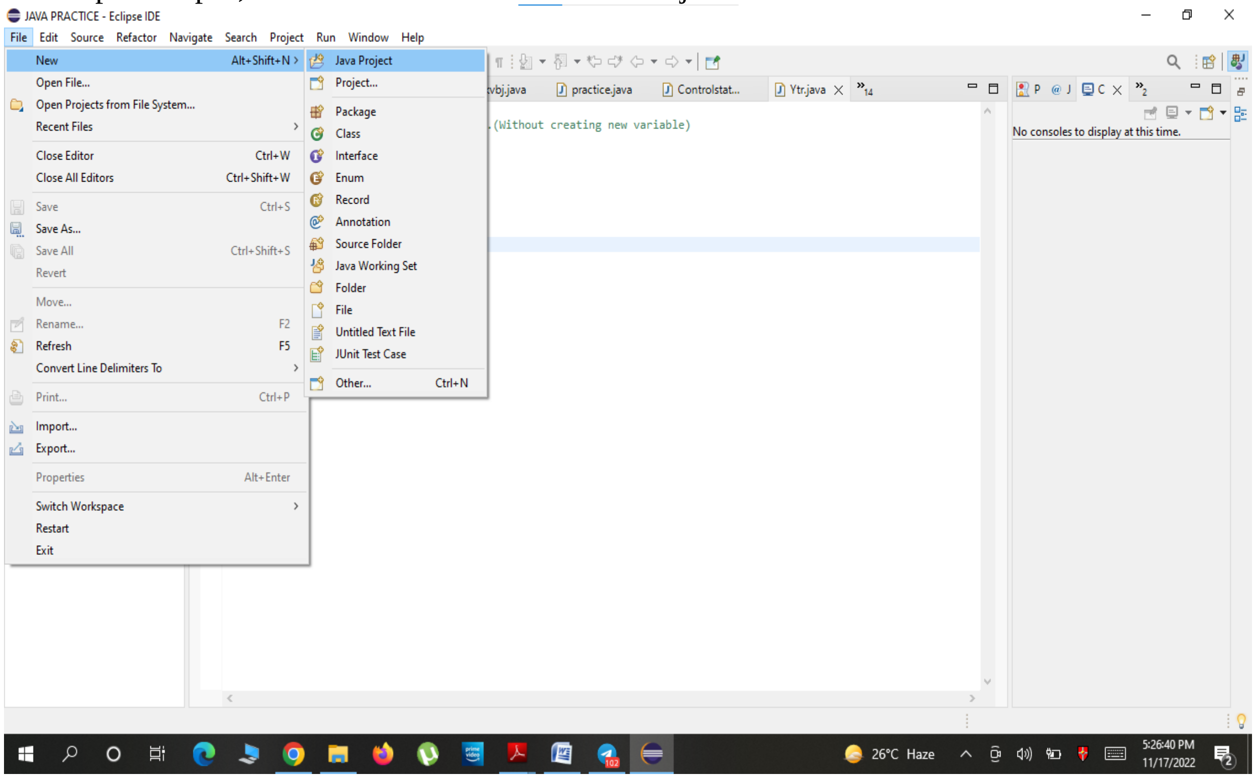
How to create Object:

```
Example obj=new Example( );
Example obj2=new Example( );
```

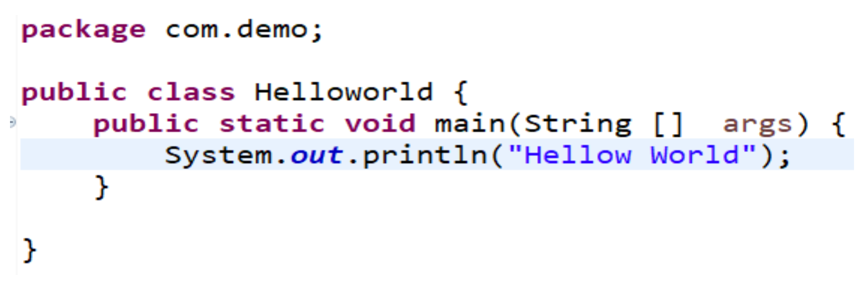
Here, obj &obj2 are objects.

Java First Program on Eclipse:

1. Install JDK, eclipse in your machine.
2. Open eclipse, click on File > > New>>Java Project



3. Give a project name than click on next. Then click on finish.
4. In project explorer, you can see your java project, right click on that project, create a new package, give valid package name e.g. com.demo. then click on finish. New package in created.
5. Right click on package, click on new>>class>>enter class name as HelloWorld>>click to finish. New class is created.
6. Write a Java program in that class as follows:

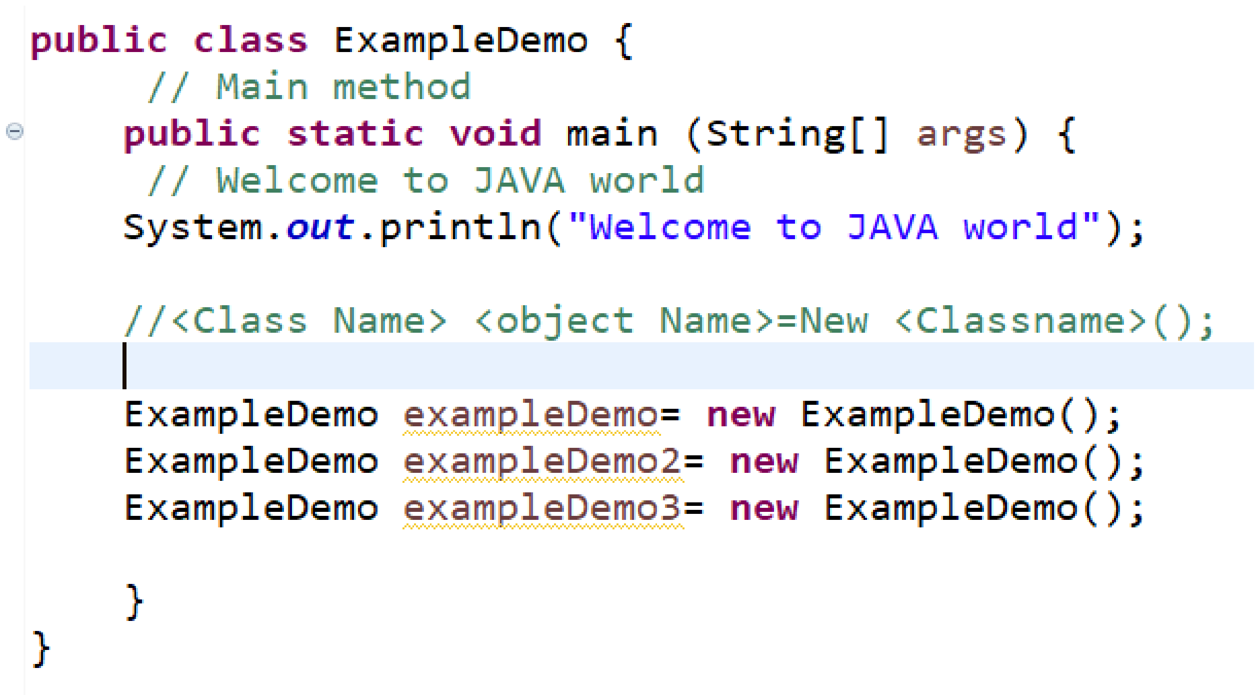


```java
package com.demo;

public class Helloworld {
    public static void main(String []  args) {
        System.out.println("Hellow World");
    }

}
```

7. To run the program, right click on that class>> run as >> java application, after clicking.output will be printed in console.



```
public class ExampleDemo {
     // Main method
    public static void main (String[] args) {
     // Welcome to JAVA world
    System.out.println("Welcome to JAVA world");

    //<Class Name> <object Name>=New <Classname>();

    ExampleDemo exampleDemo= new ExampleDemo();
    ExampleDemo exampleDemo2= new ExampleDemo();
    ExampleDemo exampleDemo3= new ExampleDemo();

    }
}
```

Identifiers:

Identifier are use for identification purpose. A name in java program is called Identifier which can be used for identification purpose. It can be method name or variable name or class name.

E.g.

```
public class Test{

        public static void main(String [] args){

                int x=10;

}

}
```

Identifiers in above e.g. are Test, main, String, args. x.

Rules for defining java identifier:

- The only allowed characters in Java identifier are as follow:
  Alphabets: A,B,C…..Z
  a,b,c….z
  Digits: 0,1,2,3,….9
  Special Symbol:~!@#$%^&*()_-+ = { }[ ] \ |< > ? / : ; " '

- Identifiers can't start with digit.
  e.g.,abc123 is valid, but 123abc is invalid.

- Java identifiers are case sensitive.
  e.g.
  ```
  public class Test {
  int digit=10:
  int Digit=10
  }
  ```

- We can't use reserved words/ Keywords as identifiers.
  ```
  int x=10: // Valid
  int if=20: // invalid
  ```

Keywords:

Keywords are reserved names whose meaning is already set at memory called as keyword

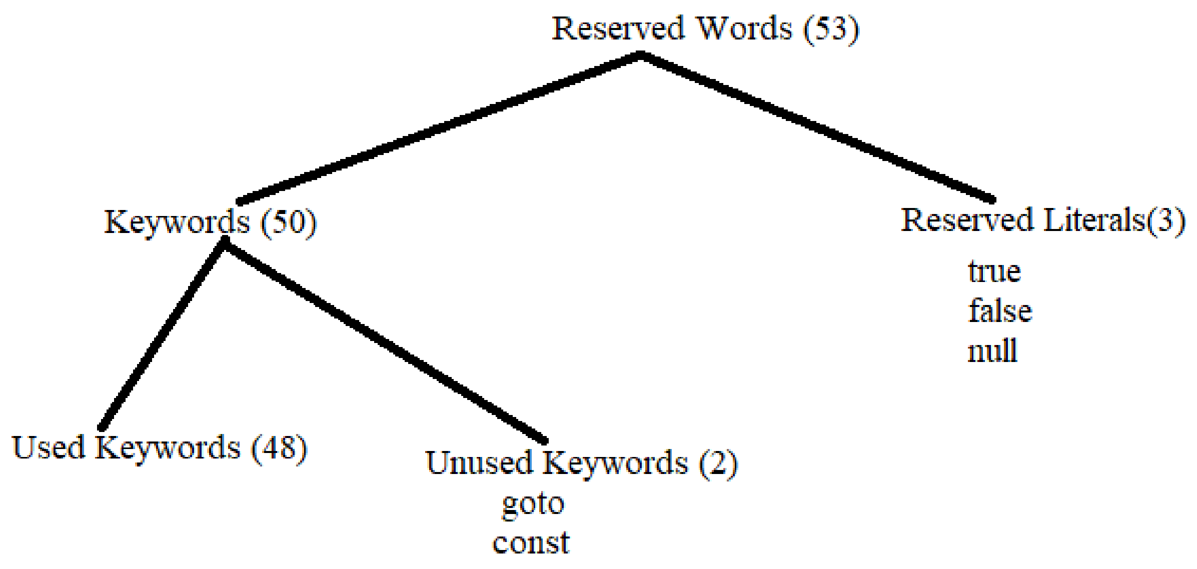


Figure Reserved Words

Flow Control	**Modifiers**	**Data Type**	**Exception Handling**	**Class Related**	**Other Keywords**
if	public	byte	throw	class	new
else	private	short	throws	package	instanceof
switch	protected	int	try	import	super
case	static	long	catch	extends	this
default	final	float	finally	implements	void
for	abstract	double	assert	interface	enum
do	synchronized	char			
while	native	boolean			
break	transient				
continue	volatile				
return	strictfp				

Data Types:

1. Data type are used to represent the type of data stored into variable.
2. In Java, every variable & every expression has some type.
3. Each & every data type is clearly defined. Every assignment should be checked by compiler for type compatibility.

There are two types if data types:

1. Primitive.
2. Non- Primitive.

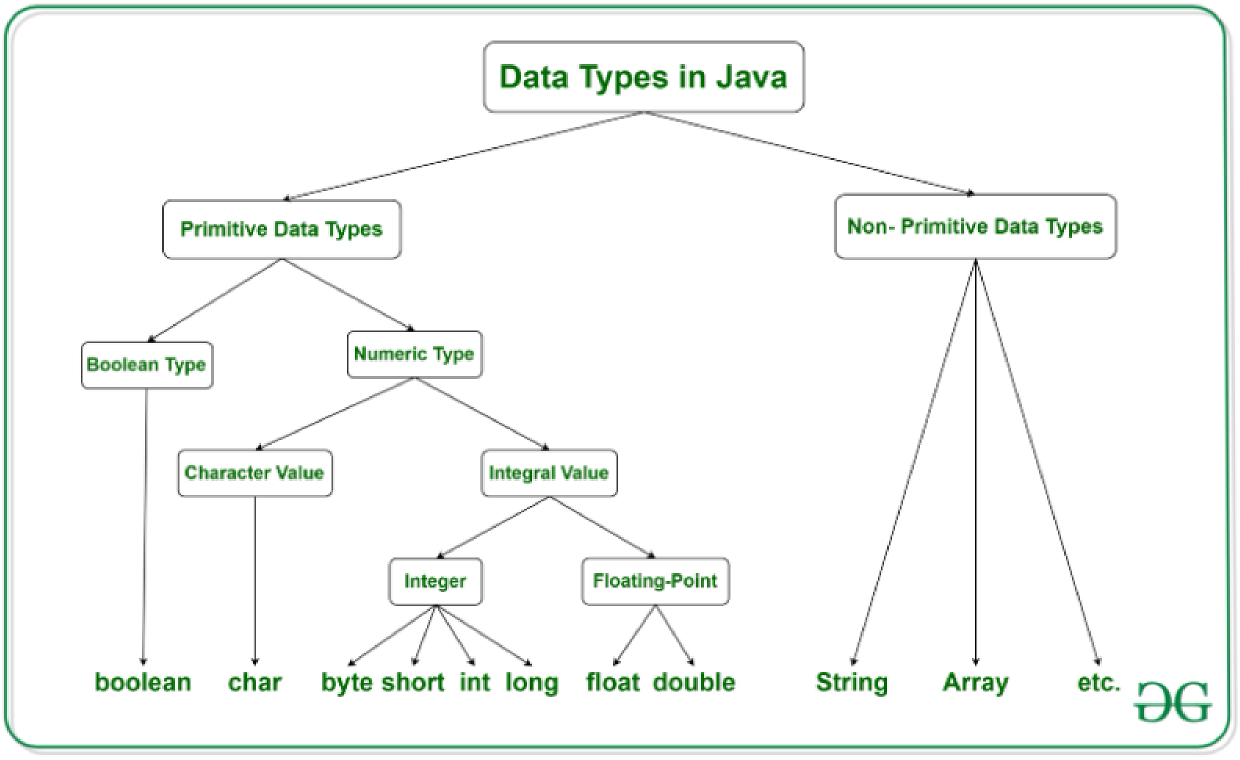


Figure: Classification of Data Type.

1) Primitive data types:

a) Numeric data types.
b) Non-Numeric data types.

a) Numeric data types:
   i. Integral data types:
      1. byte
      2. short
      3. int
      4. long

   ii. Floating Point data types:
      1. float
      2. double

b) Non-numeric data types:
   1. char
   2. boolean

Note:
Except char & boolean, remaining data types are considered as signed data types. Because, we can represent both positive & negative number.

2) Non-Primitive data type:

- Class
- Array
- String

Integral data types:

1. byte:-
   size:- 1 byte(8 bits).
   Default value: 0.
   Range: -128 to 127.
   e.g. of byte
   byte a=10; // valid
   byte b=127; // valid

2. short:-
   Range from: -32768 to 32767
   Default size: 2 bytes(16 bits)
   Default value: 0

   short data type is best suitable for 16 bits processor like 8085 but this processors are completely outdated & hence, corresponding short type is also outdated type.
   e.g. of short:-
   short s=130: // valid

3. int:

   The most commonly used data type is int.
   Range from: -2^{31} to 2^{31}-1(-2,147,483,648 to 2,147,483,647)
   Default size:- 4 bytes (32 bits)
   Default value:- 0

   e.g. of int:
      int i=130: // valid
      int i=10.5; // not valid- errors: C.E:possible loss of precision
      int i=true; // not valid- errors: C.E:incompatible type

4. long:

   Whatever int is not enough to hold big values then we should go for long data type.
   Range from:- -2^{63} to 2^{63}-1(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
   Default size:-8 bytes
   Default value:-0

   e.g. of long:
   long l=12600*60*60624*1000;//miles

   Note:- All the above data types(byte, short, int and long) can be used to represent whole number. If we want to represent real number than we should go for floating point data types

Floating point data types:

1. float:-

   If we want 5 to 6 decimal places of accuracy than we should go for float.
   Single precision: accuracy less
   Size:-4 bytes
   Range from:- -3.4e38 to 3.4e38(e38-10^38)
   Default value:- 0.0f

   e.g.
   float x=10f; //valid
   float x=10.0f; //valid
   float x=10; //valid
   float x=true; //invalid

2. <u>double:-</u>
   If we want to 14 to 15 decimal place of accuracy than we should go for double.
   Double precision: accuracy high
   Size: 8 bytes
   Default value: 0.0d

   e.g. of double
   double x =10d; // valid
   double x =10.0d; // valid
   double x =10; // valid
   double x =true; // valid

Non-numeric data type:

1. <u>boolean:</u>
   Size: not applicable.
   Range: not applicable.(allowed values are true or false)
   Default value: false

   e.g. of boolean:
   boolean b=true; // valid
   boolean b=false; // valid
   boolean b=True; // valid
   boolean b=0; // valid

2. <u>char:</u>
   Size: 2 bytes(16 bits)
   Range from: 0 to 65535.
   char is enclosed within single quote only.

   e.g. of char
   char ch='a'; // valid
   char ch1=97; // valid
   char ch2=20.0f; // valid