WEEK -1 : Introduction to JAVA

Brief History

The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

Features of Java

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java buzzwords.

A list of the most important features of the Java language is given below:

1. **Simple**
2. **Object-Oriented**
3. **Portable**
4. **Platform independent**
5. **Secured**
6. **Robust**
7. **Architecture neutral**
8. **Interpreted**
9. **High Performance**
10. **Multithreaded**
11. **Distributed**
12. **Dynamic**

Let us discuss each one of the features in detail:

1. Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

2. Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. **Object**
2. **Class**
3. **Inheritance**
4. **Polymorphism**
5. **Abstraction**
6. **Encapsulation**

3. Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

4. Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- **No explicit pointer**
- **Java Programs run inside a virtual machine sandbox**
- **Classloader:** Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically.
- **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access rights to objects.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

5. Robust

The English mining of Robust is strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

6. Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

7. Portable

Java is portable because it facilitates you to carry the Java **bytecode** to any platform. It doesn't require any implementation.

8. High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.

9. Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

10. Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads.

The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

11. Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).

JAVA Architecture

Java Architecture is a collection of components, i.e., **JVM, JRE,** and **JDK**. **It** integrates the process of interpretation and compilation. It defines all the processes involved in creating a Java program.

Java Architecture can be explained by using the following steps:

- There is a process of compilation and interpretation in Java.
- Java compiler converts the Java code into byte code.
- After that, the JVM converts the byte code into machine code.
- The machine code is then executed by the machine.

Components of Java Architecture

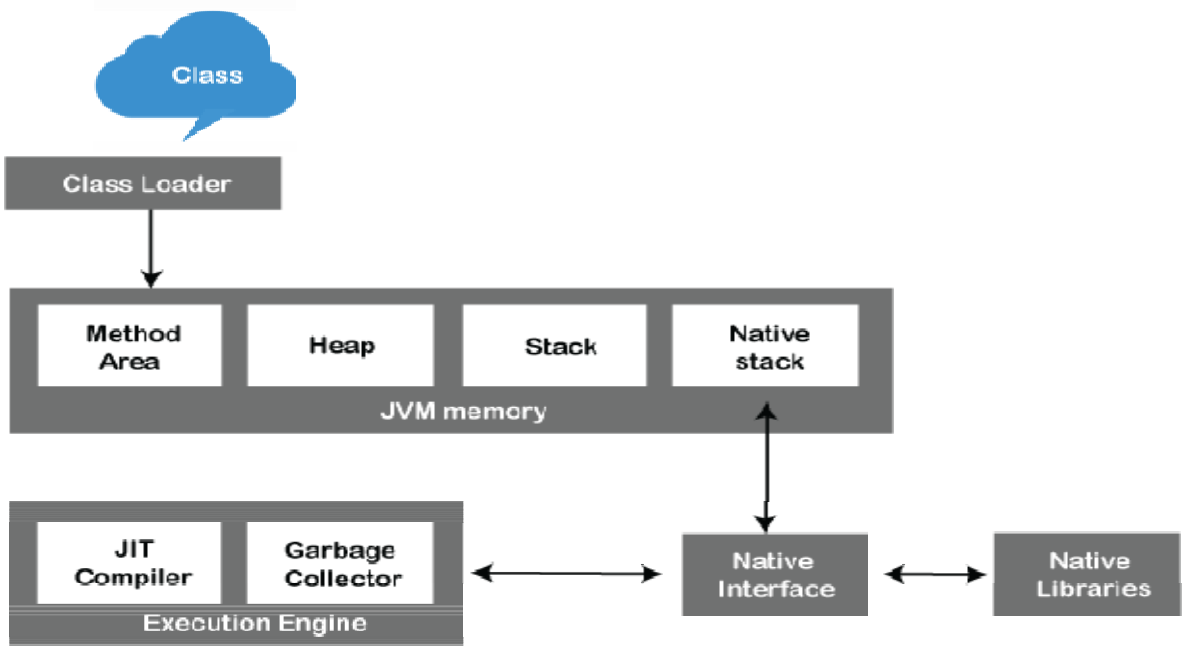
The Java architecture includes the three main components:

- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

Java Virtual Machine

The main feature of Java is **WORA**. WORA stands for **Write Once Run Anywhere**. The feature states that we can write our code once and use it anywhere or on any operating system. Our Java program can run any of the platforms only because of the Java Virtual Machine. It is a Java platform component that gives us an environment to execute java programs. JVM's main task is to convert byte code into machine code.

JVM, first of all, loads the code into memory and verifies it. After that, it executes the code and provides a runtime environment. Java Virtual Machine (JVM) has its own architecture, which is given below:



- **ClassLoader:** ClassLoader is a subsystem used to load class files. ClassLoader first loads the Java code whenever we run it.
- **Class Method Area:** In the memory, there is an area where the class data is stored during the code's execution. Class method area holds the information of static variables, static methods, static blocks, and instance methods.
- **Heap:** The heap area is a part of the JVM memory and is created when the JVM starts up. Its size cannot be static because it increase or decrease during the application runs.
- **Stack:** It is also referred to as thread stack. It is created for a single execution thread. The thread uses this area to store the elements like the partial result, local variable, data used for calling method and returns etc.
- **Native Stack:** It contains the information of all the native methods used in our application.
- **Execution Engine:** It is the central part of the JVM. Its main task is to execute the byte code and execute the Java classes. The execution engine has three main components used for executing Java classes.

  - **Interpreter:** It converts the byte code into native code and executes. It sequentially executes the code. The interpreter interprets continuously and even the same method multiple times. This reduces the performance of the system, and to solve this, the JIT compiler is introduced.
  - **JIT Compiler:** JIT compiler is introduced to remove the drawback of the interpreter. It increases the speed of execution and improves performance.
  - **Garbage Collector:** The garbage collector is used to manage the memory, and it is a program written in Java. It works in two phases, i.e., **Mark** and **Sweep**. Mark is an area where the garbage collector identifies the used and unused chunks of memory. The Sweep removes the identified object from the **Mark**
  - **Java Native Interface**

Java Native Interface works as a mediator between Java method calls and native libraries.

Java Runtime Environment

It provides an environment in which Java programs are executed. JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it

Java Development Kit

It is a software development environment used in the development of Java applications and applets. Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it. To learn more about the Java Development Kit

Applications of Java Programming Language

The expansion of the Java programming language is very wide and it is proved by the statement **3 Billion Devices Runs Java**.

Java provides a rich and wide range of API that helps programmers to develop applications. Using Java, we can develop different applications for different purposes. We can use Java technology to develop the following applications:

- Mobile App Development
- Desktop GUI Applications
- Web-based Applications
- Gaming Applications
- Big Data Technologies
- Distributed Applications
- Cloud-based Applications
- IoT Applications
- Scientific Applications
- Enterprise Applications etc..

Java Environment Setup

Now let us discuss the steps for setting up a Java environment with visual aids. Let's use the **Windows operating system** to illustrate visual aids.

Steps for setting the environment in Windows operation system are as follows:

Step 1: Java 8 JDK is available at ***https://www.oracle.com/java/technologies/downloads/#java8.*** Click the second last link for Windows(32 bit) and the last link for Windows(64 bit) as highlighted below.

Java SE Development Kit 8u121

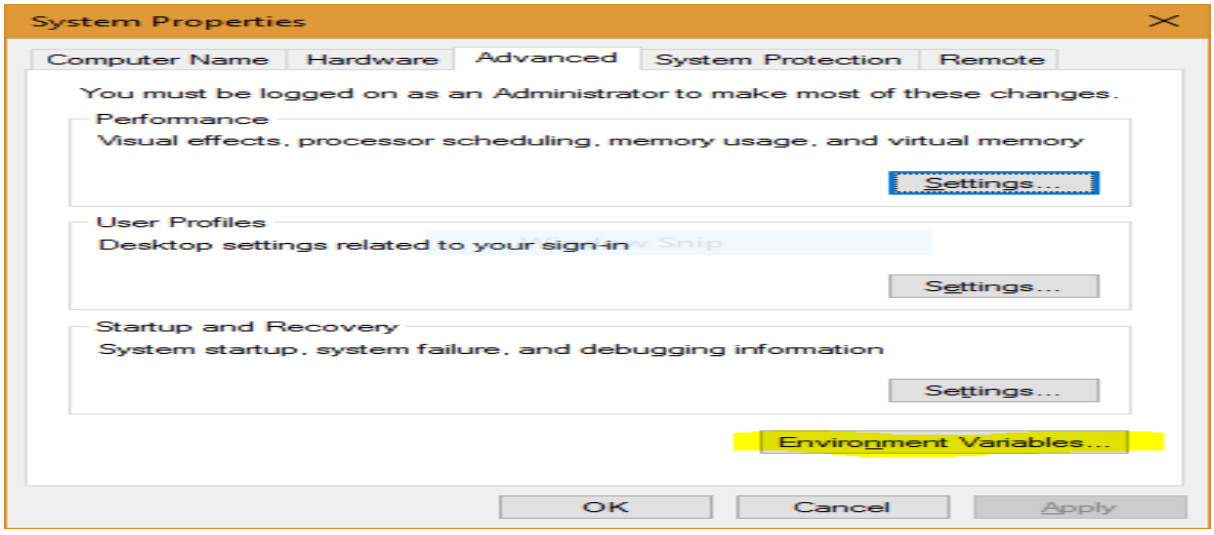
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

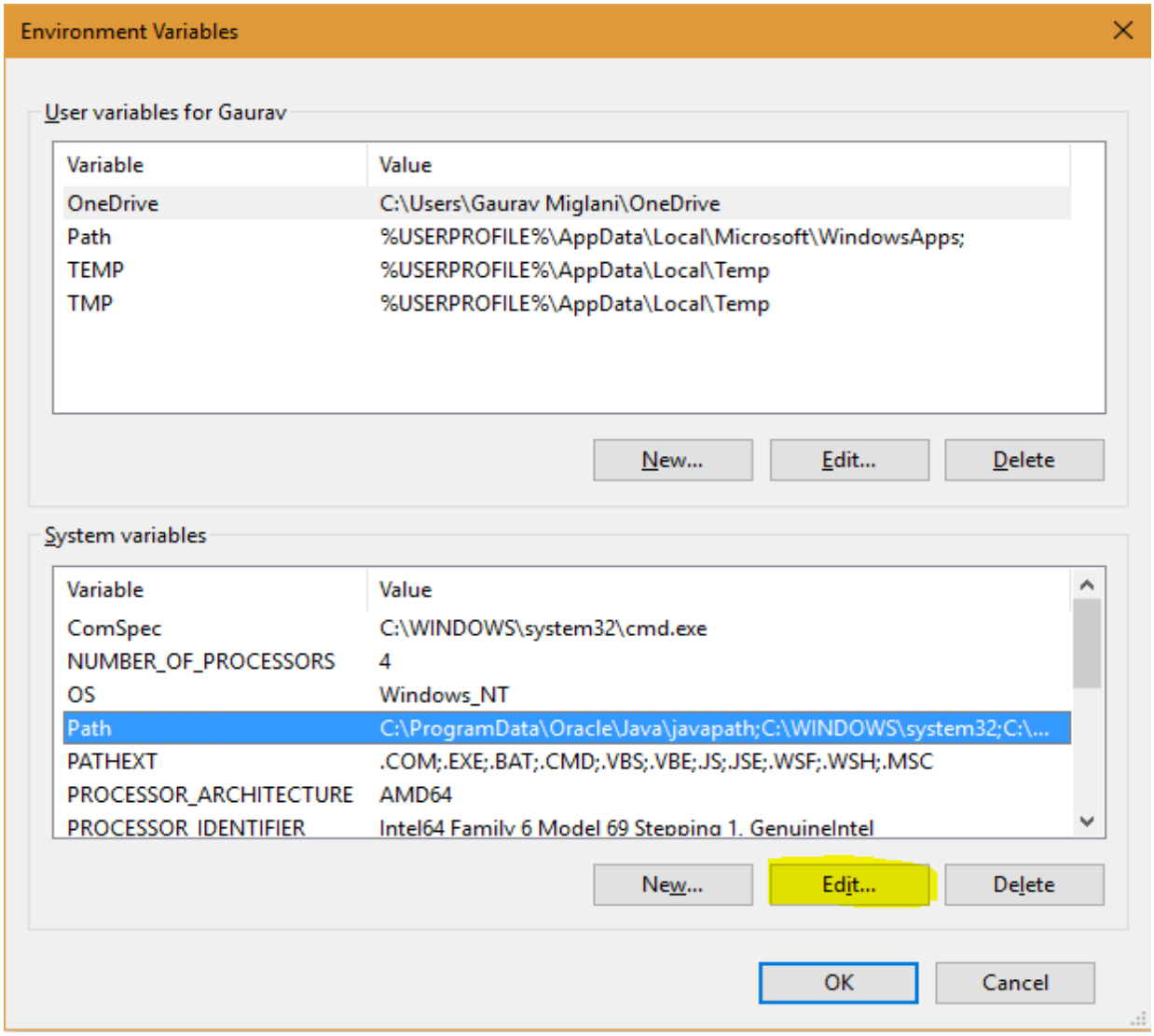
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.86 MB	jdk-8u121-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u121-linux-arm64-vfp-hflt.tar.gz
Linux x86	162.41 MB	jdk-8u121-linux-i586.rpm
Linux x86	177.13 MB	jdk-8u121-linux-i586.tar.gz
Linux x64	159.96 MB	jdk-8u121-linux-x64.rpm
Linux x64	174.76 MB	jdk-8u121-linux-x64.tar.gz
Mac OS X	223.21 MB	jdk-8u121-macosx-x64.dmg
Solaris SPARC 64-bit	139.64 MB	jdk-8u121-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.07 MB	jdk-8u121-solaris-sparcv9.tar.gz
Solaris x64	140.42 MB	jdk-8u121-solaris-x64.tar.Z
Solaris x64	96.9 MB	jdk-8u121-solaris-x64.tar.gz
Windows x86	189.36 MB	jdk-8u121-windows-i586.exe
Windows x64	195.51 MB	jdk-8u121-windows-x64.exe

Step 2: After download, run the *.exe* file and follow the instructions to install Java on your machine. Once you installed Java on your machine, you have to set up the environment variable.

Step 3: Go to **Control Panel -> System and Security -> System.** Under the Advanced System Setting option click on **Environment Variables** as highlighted below.

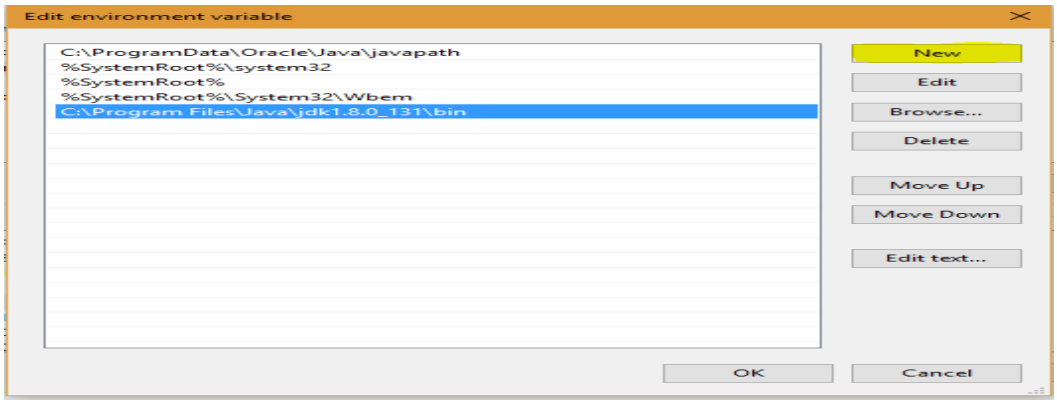


Step 4: Now, you have to alter the "Path" variable under System variables so that it also contains the path to the Java environment. Select the "Path" variable and click on the Edit button as highlighted below.



Step 5: You will see a list of different paths, click on the **New button**, and then add the path where java is installed. By default, java is installed in "**C:\Program Files\Java\jdk\bin**" folder OR "**C:\Program Files(x86)\Java\jdk\bin**".

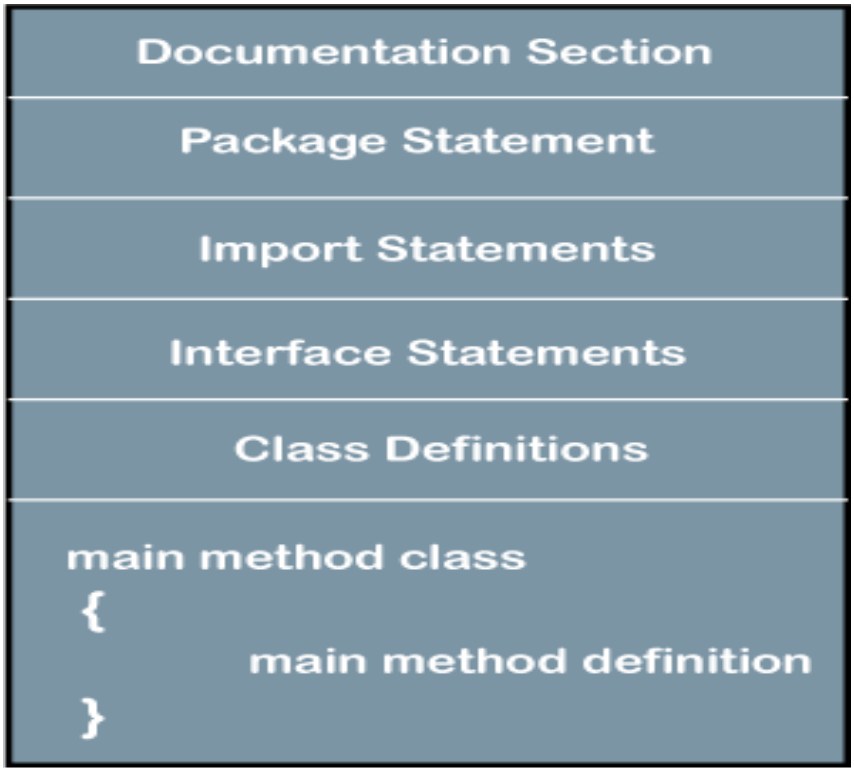
In case, you have installed java at any other location, then add that path.



Step 6: Click on OK, Save the settings, and you are done !! Now to check whether the installation is done correctly, open the command prompt and type *javac -version*. You will see that java is running on your machine.

Structure of java program

Java is an object-oriented programming, **platform-independent,** and **secure** programming language that makes it popular. Before starting the java programming lets understand the structure of java program.



Structure of Java Program

Let's see which elements are included in the structure of a Java program. A typical structure of a Java program contains the following elements:

- Documentation Section
- Package Declaration
- Import Statements
- Interface Section
- Class Definition
- Class Variables and Variables
- Main Method Class
- Methods and Behaviors

1. Documentation Section

- ✓ The documentation section is an important section but optional for a Java program. It includes **basic information** about a Java program.
- ✓ The information includes the **author's name, date of creation, version, program name, company name,** and **description** of the program.
- ✓ Whatever we write in the documentation section, the Java compiler ignores the statements during the execution of the program.
- ✓ To write the statements in the documentation section, we use **comments**. The comments may be **single-line, multi-line,** and **documentation** comments.

- **Single-line Comment:** It starts with a pair of forwarding slash **(//)**. For example:

```
//It is an example of single line comment
```

- **Multi-line Comment:** It starts with a **/*** and ends with ***/.** We write between these two symbols. For example:

```
/*It is an example of multiline comment*/
```

Documentation Comment: It starts with the delimiter **(/**)** and ends with ***/**. For example:

```
/**It is an example of documentation comment*/
```

2 . Package Declaration

The package declaration is optional. It is placed just after the documentation section. In this section, we declare the **package name** in which the class is placed.

```
package javatpoint; //where javatpoint is the package name
pckage com.javatpoint; //where com is the root directory and javatpoint is the subdirectory
```

3. Import Statements

The package contains the many predefined classes and interfaces. If we want to use any class of a particular package, we need to import that class. The import statement represents the class stored in the other package. We use the **import** keyword to import the class. It is written before the class declaration and after the package statement.

```
import java.util.Scanner; //it imports the Scanner class only
import java.util.*; //it imports all the class of the java.util package
```

4. Interface Section

It is an optional section. We can create an **interface** in this section if required. We use the **interface** keyword to create an interface. An **interface** is a slightly different from the class. It contains only **constants** and **method** declarations.

```
interface car
{
void start();
void stop();
}
```

5. Class Definition

In this section, we define the class. It is **vital** part of a Java program. Without the class, we cannot create any Java program.

- ✓ A Java program may conation more than one class definition. We use the **class** keyword to define the class.
- ✓ The class is a blueprint of a Java program. It contains information about user-defined methods, variables, and constants.
- ✓ Every Java program has at least one class that contains the main() method. For example:

```
class Student //class definition
{
}
```

6. Class Variables and Constants

In this section, we define variables and **constants** that are to be used later in the program.

- ✓ In a Java program, the variables and constants are defined just after the class definition.
- ✓ The variables and constants store values of the parameters. It is used during the execution of the program.
- ✓ We can also decide and define the scope of variables by using the modifiers. It defines the life of the variables. For example:

```
class Student //class definition
{
String sname;  //variable
int id;
double percentage;
}
```

7 Main Method Class

In this section, we define the **main() method.** It is essential for all Java programs. Because the execution of all Java programs starts from the main() method. In other words, it is an entry point of the class. It must be inside the class. Inside the main method, we create objects and call the methods. We use the following statement to define the main() method:

```
public static void main(String args[])
{
}
```

For example:

```
public class Student //class definition

{
public static void main(String args[])
{
//statements
}
}
```

8. Methods and behavior

In this section, we define the functionality of the program by using the methods. The methods are the set of instructions that we want to perform. These instructions execute at runtime and perform the specified task. For example:

```
public class Demo //class definition
{
public static void main(String args[])
{
void display()
{
System.out.println("Welcome to javatpoint");
}
//statements
}
}
```

Compilation and execution of java program

Let us look at a simple code first that will print the words **Hello World**.

Example

```
public class MyFirstJavaProgram {

  /* This is my first java program.
  * This will print 'Hello World' as the output
  */
  public static void main(String []args)
{
     System.out.println("Hello World"); // prints Hello World
  }
}
```

Let's look at how to save the file, compile, and run the program. Please follow the subsequent steps −

- Open notepad and add the code as above.
- Save the file as: MyFirstJavaProgram.java.
- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
- Type 'javac MyFirstJavaProgram.java' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line

  (Assumption: The path variable is set).
- Now, type ' java MyFirstJavaProgram ' to run your program.
- You will be able to see ' Hello World ' printed on the window.

Output

```
C:\> javac  MyFirstJavaProgram.java
C:\> java MyFirstJavaProgram
    Hello World
```

Clean coding in java

What is a clean code?

"The code that is easily readable, modifiable, maintainable and is less prone to errors is referred as Clean Code".

- Writing clean code reflects programmer's capability of writing the code that makes his/her and others programmer's lives easier.

How to write Clean Code in Java?

Java is one of the most used programming languages. Many big tech companies use Java for their operations. The points mentioned below are very important to achieve clean coding in Java.