What is Java?

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

**Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform

**Public class** Simple{

**public** **static** **void** main(String args[]){

     System.out.println("Hello Java");

    }

}

## Types of Java Applications

#### **Standalone Application**

#### **Web Application**

#### **Enterprise Application**

#### **Mobile Application**

[**James Gosling**](https://www.javatpoint.com/james-gosling-father-of-java)**, Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

## Popular Java Editors

To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following −

* **Notepad** − On Windows machine, you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
* **Netbeans** − A Java IDE that is open-source and free which can be downloaded from <https://www.netbeans.org/index.html>.
* **Eclipse** − A Java IDE developed by the eclipse open-source community and can be downloaded from <https://www.eclipse.org/>.

# **Features of Java**

The primary objective of [Java programming](https://www.javatpoint.com/java-tutorial)

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

.Simple

Object-Oriented

Portable

Platform independent

Secured

Robust

Interpreted

High Performance

Multithreaded

[Distributed](https://www.javatpoint.com/features-of-java" \l "Distributed)

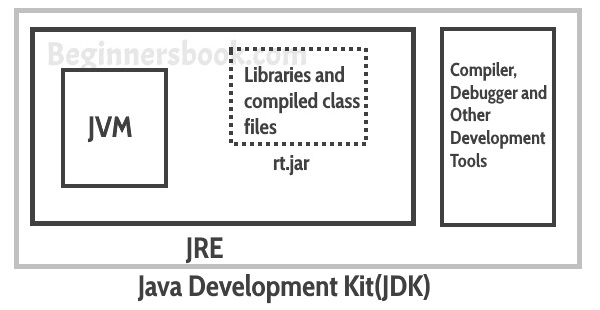
[Dynamic](https://www.javatpoint.com/features-of-java#Dynamic)

Installation process:

## JDK (Java Development Kit)

JDK is a superset of JRE, it contains everything that JRE has along with development tools such as compiler, debugger etc.

1. JDK stands for Java Development Kit. To install Java on your system, you need to first install JDK on your system.
2. JDK contains the tools that you need, in order to write and execute[java programs](https://beginnersbook.com/2017/09/java-examples/).
3. JDK consists of Private JRE, JVM, compiler, Java application launcher, Appletviewer, etc. It also provides the Standard Edition (SE) of java API.



### Features of JDK

1. The JDK comes with a complete Java Runtime Environment (JRE), that is different from the regular JRE thats the reason it is usually called a private runtime so we can say that it includes all the features that JRE has.
2. It has all the java development tools such as compiler, JVM, JRE, debugger etc.
3. You need jdk in order to write and run java program.
4. JDK supports multiple platforms and can be installed on Windows, Mac and other operating systems.

## JRE (Java Runtime Environment)

JRE is the environment within which the java virtual machine runs. JRE contains Java virtual Machine(JVM), class libraries, and other files excluding development tools such as compiler and debugger.  
Which means you can run the code in JRE but you can’t develop and compile the code in JRE.

1. JRE stands for Java Runtime Environment. It provides runtime environment for java applications.
2. You need JRE in order to run java programs. If you are not a developer and not writing java programs, you do not need JDK but you still need JRE to run java programs.
3. All the JDK already comes with JRE so you do not need to download and install it separately.
4. JRE contains set of libraries and other files that JVM uses at runtime.

## JRE

### Features of JRE

1. JRE contains set of libraries and other tools that JVM needs at runtime.
2. You can easily run any java program on JRE but you need jdk to write and compile java programs.
3. JRE contains libraries that are required for integrations such as Java Database Connectivity (JDBC), Java Naming and Directory Interface (JNDI), Remote Method Invocation (RMI) etc.

## JVM (Java Virtual Machine)

Java is a high level programming language. A program written in high level language cannot be run on any machine directly. First, it needs to be translated into that particular machine language. The **javac compiler** does this thing, it takes java program (.java file containing source code) and translates it into machine code (referred as byte code or .class file).

[Java Virtual Machine (JVM)](https://beginnersbook.com/2013/05/jvm/) is a virtual machine that resides in the real machine (your computer) and the **machine language for JVM is byte code**. This makes it easier for compiler as it has to generate byte code for JVM rather than different machine code for each type of machine. JVM executes the byte code generated by compiler and produce output. **JVM is the one that makes java platform independent**.

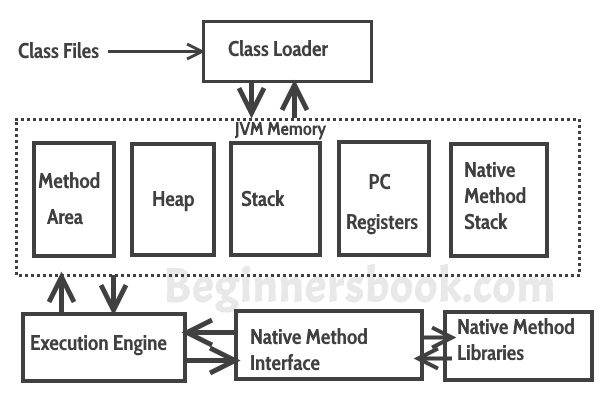
### Features of JVM

1. JVM makes it possible to run java code on any machine, it is the JVM that makes java truly platform independent.
2. It also allows to run java applications on cloud platforms.
3. JDK and JRE both of these contain JVM.
4. JVM is an interpreter as it executes the java code line by line.
5. JVM converts the bytecode into machine code. JVM is platform independent as JVM doesn’t depend on the hardware and operating system of the machine.

### How JVM works?

## JVM

### JVM Architecture



**Lets see how JVM works**:  
**Class Loader:** The class loader reads the .class file and save the byte code in the **method area**.

**Class/Method Area**: There is only one method area in a JVM which is shared among all the classes. This holds the class level information of each .class file.

**Heap**: Heap is a part of JVM memory where objects are allocated. JVM creates a Class object for each .class file.

**Stack**: Stack is a also a part of JVM memory but unlike Heap, it is used for storing temporary variables.

**PC Registers**: This keeps the track of which instruction has been executed and which one is going to be executed. Since instructions are executed by threads, each thread has a separate PC register.

**Native Method stack:** A native method can access the runtime data areas of the virtual machine.

**Native Method interface**: It enables java code to call or be called by native applications. Native applications are programs that are specific to the hardware and OS of a system.

**Garbage collection**: A class instance is explicitly created by the java code and after use it is automatically destroyed by garbage collection for memory management.

## Difference between JDK, JRE and JVM

| **JDK** | **JRE** | **JVM** |
| --- | --- | --- |
| JDK stands for Java Development Kit | JRE stands for Java Runtime Environment | JVM stands for Java Virtual Machine. |
| It contains everything that JRE has along with development tools such as compiler, debugger etc. | JRE contains Java virtual Machine(JVM), class libraries, and other files excluding development tools such as compiler and debugger. | Java Virtual Machine (JVM) is a virtual machine that resides in the real machine (your computer) and the **machine language for JVM is byte code** |
| JDK is a superset of JRE | It is a subset of JDK. | JVM is a subset of JRE. |
| JDK is used to create and compile Java programs. | JRE is a part of JDK that contains JVM. | JVM is used to run the java code on any machine. |

## Parameters used in First Java Program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

**class** Simple{

**public** **static** **void** main(String args[]){

     System.out.println("Hello Java");

    }

}

* **class** keyword is used to declare a class in Java.
* **public** keyword is an access modifier that represents visibility. It means it is visible to all.
* **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
* **void** is the return type of the method. It means it doesn't return any value.
* **main** represents the starting point of the program.
* **String[] args** or **String args[]** is used for [command line argument](https://www.javatpoint.com/command-line-argument). We will discuss it in coming section.
* **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class.