What is Java?

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

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

## **Platform**: Any hardware or software environment in which a program runs, is known as a platforJava Example

Let's have a quick look at Java programming example. A detailed description of Hello Java example is available in next page.

**Simple.java**

1. **class** Simple{
2. **public** **static** **void** main(String args[]){
3. System.out.println("Hello Java");
4. }
5. }

m. Since Java has a runtime environment (JRE) and API, it is called a platform.

## Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

#### **1) Standalone Application**

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

#### **2) Web Application**

An application that runs on the server side and creates a dynamic page is called a web application. Currently, [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), [Struts](https://www.javatpoint.com/struts-2-tutorial), [Spring](https://www.javatpoint.com/spring-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial), [JSF](https://www.javatpoint.com/jsf-tutorial), etc. technologies are used for creating web applications in Java.

#### **3) Enterprise Application**

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, [EJB](https://www.javatpoint.com/ejb-tutorial) is used for creating enterprise applications.

#### **4) Mobile Application**

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

## Java Platforms / Editions

There are 4 platforms or editions of Java:

#### **1) Java SE (Java Standard Edition)**

It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, [String](https://www.javatpoint.com/java-string), Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

#### **2) Java EE (Java Enterprise Edition)**

It is an enterprise platform that is mainly used to develop web and enterprise applications. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, [JPA](https://www.javatpoint.com/jpa-tutorial), etc.

#### **3) Java ME (Java Micro Edition)**

It is a micro platform that is dedicated to mobile applications.

#### **4) JavaFX**

It is used to develop rich internet applications. It uses a lightweight user interface API.

Release Versions in java:

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
7. J2SE 5.0 (30th Sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 (18th Mar 2014)
11. Java SE 9 (21st Sep 2017)
12. Java SE 10 (20th Mar 2018)
13. Java SE 11 (September 2018)
14. Java SE 12 (March 2019)
15. Java SE 13 (September 2019)
16. Java SE 14 (Mar 2020)
17. Java SE 15 (September 2020)
18. Java SE 16 (Mar 2021)
19. Java SE 17 (September 2021)
20. Java SE 18 (to be released by March 2022)

# **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.



1. [Simple](https://www.javatpoint.com/features-of-java#Simple)
2. [Object-Oriented](https://www.javatpoint.com/features-of-java#Object-Oriented)
3. [Portable](https://www.javatpoint.com/features-of-java#Portable)
4. [Platform independent](https://www.javatpoint.com/features-of-java#Platform-independent)
5. [Secured](https://www.javatpoint.com/features-of-java#Secured)
6. [Robust](https://www.javatpoint.com/features-of-java#Robust)
7. [Architecture neutral](https://www.javatpoint.com/features-of-java#Architecture-neutral)
8. [Interpreted](https://www.javatpoint.com/features-of-java#Interpreted)
9. [High Performance](https://www.javatpoint.com/features-of-java#High-Performance)
10. [Multithreaded](https://www.javatpoint.com/features-of-java#Multithreaded)
11. [Distributed](https://www.javatpoint.com/features-of-java#Distributed)
12. [Dynamic](https://www.javatpoint.com/features-of-java#Dynamic)

### **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.

### **Object-oriented**

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) 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](https://www.javatpoint.com/object-and-class-in-java)
2. [Class](https://www.javatpoint.com/object-and-class-in-java#class)
3. [Inheritance](https://www.javatpoint.com/inheritance-in-java)
4. [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
5. [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
6. [Encapsulation](https://www.javatpoint.com/encapsulation)

### **Platform Independent**



Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), 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.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).

### **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**



Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

### **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.

### **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.

### **Portable**

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

### **High-performance**

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

### **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.

### **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.

### **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++.

# **First Java Program | Hello World Example**

1. [Software Requirements](https://www.javatpoint.com/simple-program-of-java#hellojavareq)
2. [Creating Hello Java Example](https://www.javatpoint.com/simple-program-of-java#hellojavaex)
3. [Resolving javac is not recognized](https://www.javatpoint.com/simple-program-of-java#hellojavawhatjavacnot)

In this section, we will learn how to write the simple program of Java. We can write a simple hello Java program easily after installing the JDK.

To create a simple Java program, you need to create a class that contains the main method. Let's understand the requirement first.

### **The requirement for Java Hello World Example**

For executing any Java program, the following software or application must be properly installed.

* Install the JDK if you don't have installed it, [download the JDK](http://www.oracle.com/technetwork/java/javase/downloads/index.html) and install it.
* Set path of the jdk/bin directory. [http://www.javatpoint.com/how-to-set-path-in-java](https://www.javatpoint.com/how-to-set-path-in-java)
* Create the Java program
* Compile and run the Java program

### **Creating Hello World Example**

Let's create the hello java program:

1. **class** Simple{
2. **public** **static** **void** main(String args[]){
3. System.out.println("Hello Java");
4. }
5. }

[**Test it Now**](https://www.javatpoint.com/opr/test.jsp?filename=Simple)

Save the above file as Simple.java.

|  |  |
| --- | --- |
| **To compile:** | javac Simple.java |
| **To execute:** | java Simple |

**Output:**

Hello Java

**Compilation Flow:**

When we compile Java program using javac tool, the Java compiler converts the source code into byte code.

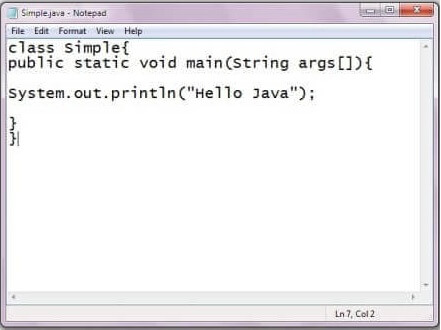


## Parameters used in First Java Program

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

* **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. We will discuss the internal working of [System.out.println()](https://www.javatpoint.com/system-out-println-in-java) statement in the coming section.

To write the simple program, you need to open notepad by **start menu -> All Programs -> Accessories -> Notepad** and write a simple program as we have shownbelow:



As displayed in the above diagram, write the simple program of Java in notepad and saved it as Simple.java. In order to compile and run the above program, you need to open the command prompt by **start menu -> All Programs -> Accessories -> command prompt**. When we have done with all the steps properly, it shows the following output:



To compile and run the above program, go to your current directory first; my current directory is c:\new. Write here:

|  |  |
| --- | --- |
| **To compile:** | javac Simple.java |
| **To execute:** | java Simple |

## In how many ways we can write a Java program?

There are many ways to write a Java program. The modifications that can be done in a Java program are given below:

**1) By changing the sequence of the modifiers, method prototype is not changed in Java.**

Let's see the simple code of the main method.

1. **static** **public** **void** main(String args[])

**2) The subscript notation in the Java array can be used after type, before the variable or after the variable.**

Let's see the different codes to write the main method.

1. **public** **static** **void** main(String[] args)
2. **public** **static** **void** main(String []args)
3. **public** **static** **void** main(String args[])

**3) You can provide var-args support to the main() method by passing 3 ellipses (dots)**

Let's see the simple code of using var-args in the main() method. We will learn about var-args later in the Java New Features chapter.

1. **public** **static** **void** main(String... args)

**4) Having a semicolon at the end of class is optional in Java.**

Let's see the simple code.

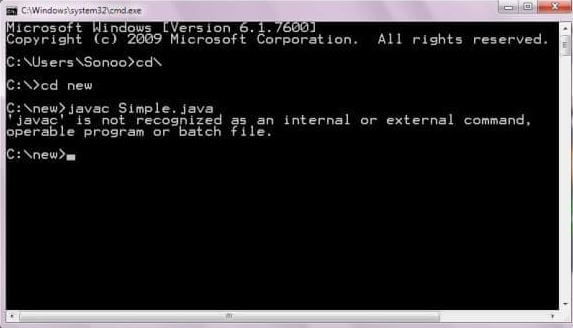
1. **class** A{
2. **static** **public** **void** main(String... args){
3. System.out.println("hello java4");
4. }
5. };

## Valid Java main() method signature

1. **public** **static** **void** main(String[] args)
2. **public** **static** **void** main(String []args)
3. **public** **static** **void** main(String args[])
4. **public** **static** **void** main(String... args)
5. **static** **public** **void** main(String[] args)
6. **public** **static** **final** **void** main(String[] args)
7. **final** **public** **static** **void** main(String[] args)
8. **final** **strictfp** **public** **static** **void** main(String[] args)

### **Resolving an error "javac is not recognized as an internal or external command"?**

If there occurs a problem like displayed in the below figure, you need to set a path. Since DOS doesn't recognize javac and java as internal or external command. To overcome this problem, we need to set a path. The path is not required in a case where you save your program inside the JDK/bin directory. However, it is an excellent approach to set the path. Click here for [How to set path in java](https://www.javatpoint.com/how-to-set-path-in-java).



# **Internal Details of Hello Java Program**

## What happens at compile time?

At compile time, the Java file is compiled by Java Compiler (It does not interact with OS) and converts the Java code into bytecode.



## What happens at runtime?

At runtime, the following steps are performed:



**Classloader:** It is the subsystem of JVM that is used to load class files.

24.3M

457

Prime Ministers of India | List of Prime Minister of India (1947-2020)

**Bytecode Verifier:** Checks the code fragments for illegal code that can violate access rights to objects.

**Interpreter:** Read bytecode stream then execute the instructions.

### **Q) Can you save a Java source file by another name than the class name?**

Yes, if the class is not public. It is explained in the figure given below:

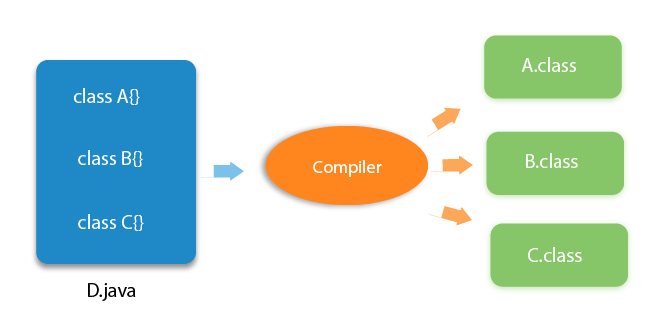


|  |  |
| --- | --- |
| **To compile:** | javac Hard.java |
| **To execute:** | java Simple |

Observe that, we have compiled the code with file name but running the program with class name. Therefore, we can save a Java program other than class name.

### **Q) Can you have multiple classes in a java source file?**

Yes, like the figure given below illustrates:



# **How to set path in Java**

1. [How to set the path of JDK in Windows OS](https://www.javatpoint.com/how-to-set-path-in-java)
   1. [Setting Temporary Path of JDK](https://www.javatpoint.com/how-to-set-path-in-java#pathtemporary)
   2. [Setting Permanent Path of JDK](https://www.javatpoint.com/how-to-set-path-in-java#pathpermanent)
2. [How to set the path of JDK in Linux OS](https://www.javatpoint.com/how-to-set-path-in-java#pathlinux)

The path is required to be set for using tools such as javac, java, etc.

If you are saving the Java source file inside the JDK/bin directory, the path is not required to be set because all the tools will be available in the current directory.

However, if you have your Java file outside the JDK/bin folder, it is necessary to set the path of JDK.

There are two ways to set the path in Java:

35.9M

597

Hello Java Program for Beginners

1. Temporary
2. Permanent

## 1) How to set the Temporary Path of JDK in Windows

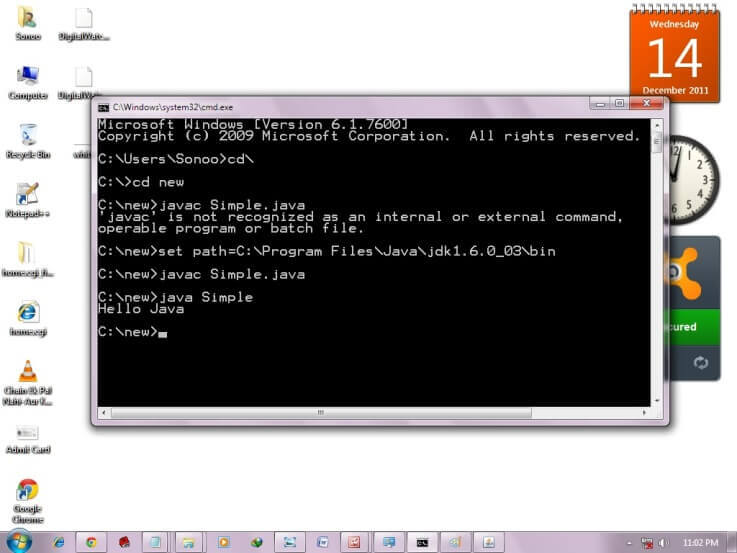
To set the temporary path of JDK, you need to follow the following steps:

* Open the command prompt
* Copy the path of the JDK/bin directory
* Write in command prompt: set path=copied\_path

### **For Example:**

set path=C:\Program Files\Java\jdk1.6.0\_23\bin

Let's see it in the figure given below:



## 2) How to set Permanent Path of JDK in Windows

For setting the permanent path of JDK, you need to follow these steps:

* Go to MyComputer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok

### **For Example:**

|  |
| --- |
| **1) Go to MyComputer properties** |
| how to set path in java |
| **2) Click on the advanced tab** |
| how to set path in java |
| **3) Click on environment variables** |
| how to set path in java |
| **4) Click on the new tab of user variables** |
| how to set path in java |
| **5) Write the path in the variable name** |
| how to set path in java |
| **6) Copy the path of bin folder** |
| how to set path in java |
| **7) Paste path of bin folder in the variable value** |
| how to set path in java |
| **8) Click on ok button** |
| how to set path in java |
| **9) Click on ok button** |
| how to set path in java |

Now your permanent path is set. You can now execute any program of java from any drive.

### **Setting Java Path in Linux OS**

Setting path in Linux OS is the same as setting the path in the Windows OS. But, here we use the export tool rather than set. Let's see how to set path in Linux OS:

export PATH=$PATH:/home/jdk1.6.01/bin/

Here, we have installed the JDK in the home directory under Root (/home).