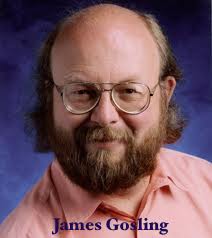
**https://www.javatpoint.com/java-basics**

# **History of Java**

**The history of Java** is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". [Java](https://www.javatpoint.com/java-tutorial) 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.



Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. There are given significant points that describe the history of Java.



1) [**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**.

2) Initially designed for small, [embedded systems](https://www.javatpoint.com/embedded-system-tutorial) in electronic appliances like set-top boxes.

3) Firstly, it was called **"Greentalk"** by James Gosling, and the file extension was .gt.

 After that, it was called **Oak** and was developed as a part of the Green project.

## Why Java named "Oak"?

5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

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

# **What is Java?**

Java is a high-level, general-purpose, object-oriented, and secure programming language developed by James Gosling at Sun Microsystems, Inc. in 1991. It is formally known as OAK. In 1995, Sun Microsystem changed the name to Java. In 2009, Sun Microsystem takeover by Oracle Corporation.

Editions of Java

Each edition of Java has different capabilities. There are three editions of Java:

* **Java Standard Editions (JSE):** It is used to create programs for a desktop computer.

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.

* **Java Enterprise Edition (JEE):** It is used to create large programs that run on the server and manages heavy traffic and complex transactions.

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

* **Java Micro Edition (JME):** It is used to develop applications for small devices such as set-top boxes, phone, and appliances. It is a micro platform which is mainly used to develop mobile applications.
* **JavaFX :**

It is used to develop rich internet applications. It uses a light-weight user interface API.

Types of Java Applications

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

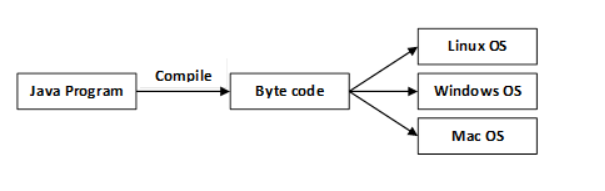
* **Standalone Applications:** Java standalone applications uses GUI components such as AWT, Swing, and JavaFX. These components contain buttons, list, menu, scroll panel, etc. It is also known as desktop alienations.
* **Enterprise Applications:** An application which is distributed in nature is called enterprise applications.
* **Web Applications:** An applications that run on the server is called web applications. We use JSP, Servlet, Spring, and Hibernate technologies for creating web applications.
* **Mobile Applications:** Java ME is a cross-platform to develop mobile applications which run across smartphones. Java is a platform for App Development in Android.

Java Platform

Java Platform is a collection of programs. It helps to develop and run a program written in the Java programming language. Java Platform includes an execution engine, a compiler and set of libraries. Java is a platform-independent language.

Features of Java

* **Simple:** Java is a simple language because its syntax is simple, clean, and easy to understand. Complex and ambiguous concepts of C++ are either eliminated or re-implemented in Java. For example, pointer and operator overloading are not used in Java.
* **Object-Oriented:** In Java, everything is in the form of the object. It means it has some data and behavior. A program must have at least one class and object.
* **Robust:** Java makes an effort to check error at run time and compile time. It uses a strong memory management system called garbage collector. Exception handling and garbage collection features make it strong.
* **Secure:** Java is a secure programming language because it has no explicit pointer and programs runs in the virtual machine. Java contains a security manager that defines the access of Java classes.
* **Platform-Independent:** Java provides a guarantee that code writes once and run anywhere. This byte code is platform-independent and can be run on any machine.



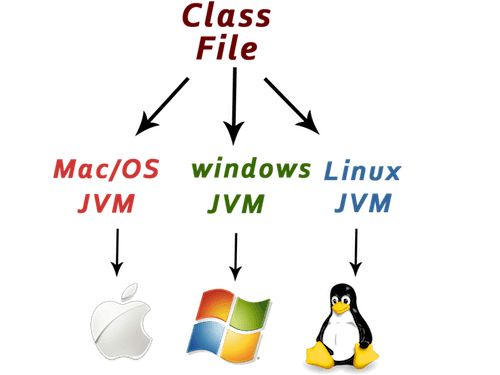
 A platform is the hardware or software environment in which a program runs.

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 the top of other hardware-based platforms. It has two components:

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

Java code can be run 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)**



* **Portable:** Java Byte code can be carried to any platform. No implementation-dependent features. Everything related to storage is predefined, for example, the size of primitive data types.
* **High Performance:** Java is an interpreted language. Java enables high performance with the use of the Just-In-Time compiler.
* **Distributed:** Java also has networking facilities. It is designed for the distributed environment of the internet because it supports TCP/IP protocol. It can run over the internet. EJB and RMI are used to create a distributed system.
* **Multi-threaded:** Java also supports multi-threading. It means to handle more than one job a time.



## OOPs (Object Oriented Programming System)

Object-oriented programming is a way of solving a complex problem by breaking them into a small sub-problem. An object is a real-world entity. It is easier to develop a program by using an object. In OOPs, we create programs using class and object in a structured manner.

**Class:** A class is a template or blueprint or prototype that defines data members and methods of an object. An object is the instance of the class. We can define a class by using the class keyword.

**Object:** An object is a real-world entity that can be identified distinctly. For example, a desk, a circle can be considered as objects. An object has a unique behavior, identity, and state. Data fields with their current values represent the state of an object (also known as its properties or attributes).

**Abstraction:** An abstraction is a method of hiding irrelevant information from the user. For example, the driver only knows how to drive a car; there is no need to know how does the car run. We can make a class abstract by using the keyword abstract. In Java, we use abstract class and interface to achieve abstraction.

**Encapsulation:** An encapsulation is the process of binding data and functions into a single unit. A class is an example of encapsulation. In Java, Java bean is a fully encapsulated class.

**Inheritance:** Inheritance is the mechanism in which one class acquire all the features of another class. We can achieve inheritance by using the extends keyword. It facilitates the reusability of the code.

**Polymorphism:** The polymorphism is the ability to appear in many forms. In other words, single action in different ways. For example, a boy in the classroom behaves like a student, in house behaves like a son. There are two types of polymorphism: run time polymorphism and compile-time polymorphism.

# **C++ vs Java**

There are many differences and similarities between the [C++ programming](https://www.javatpoint.com/cpp-tutorial) language and [Java](https://www.javatpoint.com/java-tutorial). A list of top differences between C++ and Java are given below:

|  |  |  |
| --- | --- | --- |
| **Comparison Index** | **C++** | **Java** |
| **Platform-independent** | C++ is platform-dependent. | Java is platform-independent. |
| **Mainly used for** | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| **Design Goal** | C++ was designed for systems and applications programming. It was an extension of [C programming language](https://www.javatpoint.com/c-programming-language-tutorial). | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience. |
| **Goto** | C++ supports the [goto](https://www.javatpoint.com/cpp-goto-statement) statement. | Java doesn't support the goto statement. |
| **Multiple inheritance** | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by [interfaces in java](https://www.javatpoint.com/interface-in-java). |
| **Operator Overloading** | C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading). | Java doesn't support operator overloading. |
| **Pointers** | C++ supports [pointers](https://www.javatpoint.com/cpp-pointers). You can write pointer program in C++. | Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java. |
| **Compiler and Interpreter** | C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent. | Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent. |
| **Call by Value and Call by reference** | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| **Structure and Union** | C++ supports structures and unions. | Java doesn't support structures and unions. |
| **Thread Support** | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in [thread](https://www.javatpoint.com/multithreading-in-java) support. |
| **Documentation comment** | C++ doesn't support documentation comment. | Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code. |
| **Virtual Keyword** | C++ supports virtual keyword so that we can decide whether or not override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |
| **unsigned right shift >>>** | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. |
| **Inheritance Tree** | C++ creates a new inheritance tree always. | Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the [inheritance](https://www.javatpoint.com/inheritance-in-java) tree in java. |
| **Hardware** | C++ is nearer to hardware. | Java is not so interactive with hardware. |
| **Object-oriented** | C++ is an object-oriented language. However, in C language, single root hierarchy is not possible. | Java is also an [object-oriented](https://www.javatpoint.com/java-oops-concepts) language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |

## C++ Example

File: main.cpp

1. #include <iostream>
2. **using** **namespace** std;
3. **int** main() {
4. cout << "Hello C++ Programming";
5. **return** 0;
6. }

## Java Example

File: Simple.java

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

