CoRe Java

# What is Java?

**Java** is a **programming language** and a **computing platform** that was first released by **Sun Microsystems** in 1995 (now owned by **Oracle Corporation**).

It is used to create software applications that can run on different types of devices—like computers, mobile phones, servers, and even smart cards.

## Java is known for being:

* **Object-Oriented** – everything in Java is based on objects and classes.
* **Platform-Independent** – thanks to the **Java Virtual Machine (JVM)**, Java code can run on any system that has the JVM installed, regardless of the underlying hardware or operating system.
* **Secure and Robust** – Java has strong memory management and security features.
* **Widely Used** – for web apps, Android apps, enterprise-level applications, cloud development, and more.

# **History of Java**

* **1991** – Java began as a project called **"Oak"**, developed by **James Gosling** and his team at Sun Microsystems. It was originally designed for embedded devices and set-top boxes.
* **1995** – Oak was renamed **Java**, and it was officially released to the public. It became popular for developing internet applications.
* **1998** – Java 2 was released, marking the beginning of different editions like J2SE (Standard), J2EE (Enterprise), and J2ME (Micro Edition).
* **2006** – Sun made Java open-source by releasing most of its code under the GNU General Public License (GPL).
* **2010** – Oracle Corporation acquired Sun Microsystems and took over the development of Java.
* **Present** – Java continues to evolve with regular updates, latest being **Java 21** (as of 2024), bringing performance improvements and new features.

# **Key Features of Java**

### 1. **Simple**

* Easy to learn and use.
* Removes complex features like pointers and operator overloading.

### 2. **Object-Oriented**

* Everything in Java is treated as an object.
* Supports concepts like inheritance, encapsulation, polymorphism, and abstraction.

### 3. **Platform-Independent**

* Java programs are compiled into **bytecode**, which runs on any device with a **Java Virtual Machine (JVM)**.
* This gives Java its famous **“Write Once, Run Anywhere”** capability.

### 4. **Secure**

* Java provides a secure runtime environment by managing memory access and using a security manager.
* It avoids vulnerabilities like buffer overflows.

### 5. **Robust**

* Java emphasizes error handling and memory management.
* Features like garbage collection and exception handling improve stability.

### 6. **Multithreaded**

* Java supports multiple threads of execution, allowing for efficient use of system resources.

### 7. **High Performance**

* While not as fast as low-level languages like C++, Java offers good performance due to its **Just-In-Time (JIT)** compiler.

### 8. **Distributed**

* Java has built-in features like RMI and EJB for building distributed systems (apps that run on different machines).

### 9. **Dynamic**

* Java can dynamically load classes at runtime, making it adaptable and flexible.

# **C++ vs Java**

|  |  |  |
| --- | --- | --- |
| **Feature** | **C++** | **Java** |
| **Platform Dependency** | Platform-dependent (compiled into machine code for a specific OS) | Platform-independent (compiled to bytecode, runs on JVM) |
| **Compilation** | Compiled language (uses a compiler) | Compiled and interpreted (compiled to bytecode, then interpreted by JVM) |
| **Memory Management** | Manual (uses pointers, new/delete) | Automatic (uses Garbage Collection) |
| **Pointers** | Supports pointers | No direct pointer support (for security reasons) |
| **Multiple Inheritance** | Supported through classes | Not supported through classes (only via interfaces) |
| **Speed** | Generally faster (closer to hardware) | Slower due to JVM overhead |
| **Syntax** | Complex with more features like operator overloading | Simpler and more consistent |
| **Use Cases** | System/software development, games, drivers, real-time systems | Web apps, mobile apps (Android), enterprise-level apps |
| **Object-Oriented** | Partial (supports both procedural and OOP) | Purely object-oriented (everything is in classes/objects) |
| **Standard Library** | Rich, but lower-level | Huge standard library and built-in APIs |
| **Exception Handling** | Available but less enforced | Strong exception handling built-in |
| **Security** | Less secure (direct memory access) | More secure (no pointer, sandboxing in JVM) |

### **Summary:**

* **C++** is ideal when you need high performance and low-level system control (like in game engines or operating systems).
* **Java** is preferred for cross-platform applications, especially in enterprise, web, and Android development.

## **Java Naming Conventions**

Naming conventions in Java make code **more readable**, **maintainable**, and **professional**. These are not enforced by the compiler but are strongly recommended.

### 🔷 **1. Class Names**

* **Convention**: PascalCase (each word starts with a capital letter)
* **Example**: StudentDetails, BankAccount, EmployeeManager

✅ **Tip**: Class names should be nouns.

### 🔷 **2. Interface Names**

* **Convention**: PascalCase
* **Example**: Runnable, Readable, Comparable

✅ Often describes capabilities or behaviors.

### 🔷 **3. Method Names**

* **Convention**: camelCase (starts with lowercase, next words capitalized)
* **Example**: getStudentName(), calculateTotal(), printReport()

✅ Method names should be verbs or verb phrases.

### 🔷 **4. Variable Names**

* **Convention**: camelCase
* **Example**: studentName, totalAmount, isAvailable

✅ Variable names should be meaningful and descriptive.

### 🔷 **5. Constant Names**

* **Convention**: ALL\_UPPERCASE with words separated by underscores
* **Example**: MAX\_VALUE, PI, DEFAULT\_TIMEOUT

✅ Use final keyword for constants.

### 🔷 **6. Package Names**

* **Convention**: all lowercase, may include domain-style naming
* **Example**: com.example.project, org.company.module

✅ Avoid uppercase letters and underscores.

### 🔷 **7. Enum Names**

* **Convention**: PascalCase for enum type, ALL\_UPPERCASE for enum values
* **Example**:

enum Day {

MONDAY, TUESDAY, WEDNESDAY

}

### 🔷 **8. Generic Type Parameters**

* **Convention**: Single capital letters
* **Example**: T (Type), E (Element), K (Key), V (Value)

class Box<T> {

T value;

}