



## Java – History & Complete Guide

---



### 1. History of Java

- **Early 1990s** – Sun Microsystems formed a team called the **Green Team**, led by **James Gosling**, to create a programming language for consumer electronic devices (like TVs, set-top boxes).
  - The first version was called **Oak** (named after an oak tree outside Gosling's office).
  - Oak had problems with trademark issues → renamed to **Java** (inspired by Java coffee from Indonesia).
- 
- ◆ **Java Timeline**
    - **1991** – Project “Green” initiated.
    - **1995** – Java officially released as Java 1.0 (with slogan *Write Once, Run Anywhere*).
    - **1996** – First stable release **Java 1.0.2** (widely adopted).
    - **1999** – Sun announced **J2SE, J2EE, J2ME** (Standard, Enterprise, Micro editions).
    - **2004** – Release of Java 5 (major upgrade: Generics, Annotations, Autoboxing).
    - **2006** – Sun released Java as **open source** (OpenJDK).

- **2009** – Oracle Corporation acquired Sun Microsystems → took over Java.
  - **2014 onwards** – Oracle introduced new release cycle: feature releases every 6 months.
  - **2021** – Java 17 (LTS version, widely used in enterprises).
  - **2023** – Java 21 (latest LTS version).
  - **2025** – Java remains dominant in **enterprise, Android, and DevOps ecosystems**.
- 

- ◆ **2. What is Java?**

Java is a **class-based, object-oriented programming language** designed to minimize dependencies and be **platform independent**.

- Compiled into **bytecode**, executed by the **Java Virtual Machine (JVM)**. ENV
  - Known for reliability, portability, and scalability.
- 

- ◆ **3. Key Features of Java**

1. **Platform Independent** → Runs anywhere with JVM.
2. **Object-Oriented** → Uses classes & objects.
3. **Secure** → No pointers, runs inside JVM sandbox.
4. **Robust** → Strong memory management & exception handling.

5. **Multithreading** → Runs tasks in parallel.
  6. **Portable** → Same code works on all platforms.
  7. **Automatic Garbage Collection** → Handles memory cleanup.
- 

- ◆ **4. Java Architecture**

- **Compilation:** `.java` file → compiled by `javac` → `.class` bytecode.
- **Execution:** JVM interprets bytecode → native machine code.

---

**Flow:**

Source Code → Compiler → Bytecode → JVM → Native OS

---

- ◆ **5. Editions of Java**

- **Java SE (Standard Edition)** – Core features (JVM, libraries, collections).
  - **Java EE (Enterprise Edition)** – Web/enterprise apps (Servlets, JSP, EJB).
  - **Java ME (Micro Edition)** – Mobile/embedded systems.
  - **Jakarta EE** – Successor of Java EE (modern enterprise apps).
- 

- ◆ **6. Java Basics**



**Hello World**

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, Java!");  
    }  
}
```

- `class` → Blueprint of objects.
  - `main()` → Entry point.
  - `System.out.println()` → Console output.
- 

#### ◆ 7. OOP Principles in Java

- **Encapsulation** – Wraps data & methods.
  - **Inheritance** – One class extends another.
  - **Polymorphism** – Same method, different behavior.
  - **Abstraction** – Hides implementation details.
- 

#### ◆ 8. Java Ecosystem & Tools

- **Build Tools:** Maven, Gradle
- **Frameworks:** Spring, Hibernate

- **Testing:** JUnit, TestNG
  - **DevOps Tools:** Jenkins (built in Java), Hadoop
- 

- ◆ **9. Uses of Java**

- Enterprise apps (banking, e-commerce).
  - Android app development.
  - Backend services with Spring.
  - Big Data (Hadoop ecosystem).
  - DevOps (Jenkins, Maven, Kafka).
- 

- ◆ **10. Advantages**

- ✓ Platform-independent
  - ✓ Secure & reliable
  - ✓ Strong community & ecosystem
  - ✓ Excellent for enterprise-scale apps
- 

- ◆ **11. Limitations**

- ✗ Slower than C/C++ (due to JVM).
- ✗ More verbose than Python/Kotlin.
- ✗ Higher memory consumption.

---

## ◆ 12. Future of Java

- Continues to be the **backbone of enterprise systems**.
  - Competes with Python, Go, Rust, Kotlin in specialized areas.
  - Still a **top 3 programming language worldwide** in 2025.
- 

## Summary

- **History:** Born in 1991, released in 1995, now one of the most mature languages.
  - **Strengths:** Platform-independent, robust, secure, widely used.
  - **Applications:** From Android apps → Banking → DevOps tools.
  - **Future:** Stable and essential in enterprises, cloud, and DevOps.
-