**☕ History of Java**

* **1991**: James Gosling and team at Sun Microsystems start the *Green Project* to build a language for embedded systems.
* **Originally named "Oak"**: Inspired by an oak tree outside Gosling’s office.
* **Renamed to "Java"**: After Java coffee beans—symbolizing energy and uniqueness.
* **1995**: Official launch with the slogan **“Write Once, Run Anywhere”** (WORA).
* **1997**: Java becomes a standard by ISO and ANSI.
* **1999**: Java splits into editions—J2SE (Standard), J2EE (Enterprise), J2ME (Micro).
* **2004–2014**: Major updates like Generics, Annotations, Lambda Expressions, Streams.
* **2010**: Oracle acquires Sun Microsystems.
* **2017–2025**: Modular system (Project Jigsaw), Virtual Threads, Record Patterns, and rapid 6-month release cycles.

**🔧 What Is Core Java?**

* **Core Java = Java SE (Standard Edition)**: The foundation for all other editions.
* **Focus**: Desktop applications, basic syntax, and OOP principles.
* **Includes**:
  + Java Fundamentals (data types, operators, control flow)
  + OOP Concepts (Class, Object, Inheritance, Polymorphism, Abstraction, Encapsulation)
  + Exception Handling
  + Collections Framework
  + Multithreading
  + JDBC (basic DB connectivity)
  + Swing & Applets (for GUI)
* **Used for**: Learning, interviews, and building general-purpose applications.

**🧠 Principles of Java**

**1. OOP Principles**

* **Encapsulation**: Wrap data + methods into a single unit (class).
* **Inheritance**: Reuse code by inheriting properties from parent classes.
* **Polymorphism**: One interface, many implementations.
* **Abstraction**: Hide internal details, expose only functionality.

**2. SOLID Principles (Advanced Design)**

* **S**ingle Responsibility
* **O**pen/Closed
* **L**iskov Substitution
* **I**nterface Segregation
* **D**ependency Inversion

**🖥️ What Is Java SE?**

* **Java SE = Java Standard Edition**
* Core platform for general-purpose development.
* Includes:
  + **JDK (Java Development Kit)**: Compiler, JVM, debugger, tools.
  + **Java API Libraries**: java.util, java.io, java.lang, etc.
  + **Swing, AWT**: GUI development.
  + **JDBC**: Database connectivity.
  + **Multithreading & Networking APIs**
* **Used for**: Desktop apps, command-line tools, foundational learning before moving to Java EE or Android.

**☕ Java Installation Notes**

**🔧 Step-by-Step Installation Process**

1. **Download JDK (Java Development Kit)**
   * Visit [Oracle’s official JDK page](https://www.oracle.com/java/technologies/javase-downloads.html).
   * Choose the latest stable version (e.g., JDK 17 or 21).
   * Select the installer based on your OS (Windows/macOS/Linux).
2. **Install JDK**
   * Run the installer and follow default prompts.
   * JDK typically installs in:  
     C:\Program Files\Java\jdk-<version>
3. **Set Environment Variables**
   * Open System Properties → Advanced → Environment Variables.
   * Under **System Variables**, add:
     + JAVA\_HOME = C:\Program Files\Java\jdk-<version>
     + Add %JAVA\_HOME%\bin to the Path variable.
4. **Verify Installation**
   * Open Command Prompt and run:
   * java -version
   * javac -version
   * You should see the installed version printed.

**📊 Diagram-Style Workflow (Compile & Run)**

[ Hello.java ]

↓ (javac Hello.java)

[ Hello.class ] ← Bytecode

↓ (java Hello)

[ JVM ] → [ OS ]

* **Hello.java**: Your source code.
* **javac**: Java compiler converts .java to .class.
* **Hello.class**: Bytecode file.
* **JVM (Java Virtual Machine)**: Executes bytecode.
* **OS**: JVM interacts with OS to run the program.

**🧠 Key Concepts from Installation Lecture**

* **JDK**: Includes compiler (javac), JVM, and tools.
* **JRE**: Java Runtime Environment—only for running, not compiling.
* **JVM**: Platform-independent engine that runs .class files.
* **Bytecode**: Intermediate code generated by compiler, portable across systems.

A computer program with arrows

AI-generated content may be incorrect.

**First Program:**

***public* class HelloWorld**

**{**

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

**{**

**System.*out*.println("Hello World !!!");**

**}**

**}**

**🧠 Java First Program Breakdown: HelloWorld**

**📦 class HelloWorld**

* **Definition**: A class is the basic building block in Java. It defines a blueprint for objects.
* **Keyword**: class is used to declare a class.
* **Name**: HelloWorld is the class name. By convention, it starts with a capital letter.
* **Access Specifier**: public means this class is accessible from any other class in any package.

**🔐 Access Specifiers**

| **Modifier** | **Scope of Access** |
| --- | --- |
| public | Accessible from any class anywhere |
| private | Accessible only within the same class |
| protected | Accessible within the same package + subclasses |
| *default* | Accessible only within the same package |

In your program:

* public class HelloWorld → class is globally accessible.
* public static void main → method is globally accessible and entry point for JVM.

**⚙️ static Keyword**

* **Purpose**: Belongs to the class, not to any instance.
* **Usage in main**: JVM calls main() without creating an object of the class.
* **Implication**: You can access static methods/variables directly using the class name.

**🚀 main Method**

public static void main(String[] args)

* **public**: JVM needs to access this method from outside the class.
* **static**: JVM calls it without creating an object.
* **void**: It doesn’t return any value.
* **String[] args**: Command-line arguments passed as an array of Strings.

**🖨️ System.out.println("Hello World");**

* **System**: Final class from java.lang package.
* **out**: Static member of System, an instance of PrintStream.
* **println()**: Method to print text followed by a newline.

**⚠️ Common Mistakes**

* "Hello World" must be in double quotes. Without quotes, it’s treated as a variable.
* Java is case-sensitive: Public ≠ public.
* No semicolon after closing brace }.

**Java data types** and **keywords**

**🧠 Java Data Types**

Java is **statically typed**, meaning every variable must be declared with a type. Data types are divided into two categories:

**🔹 Primitive Data Types (8 total)**

These store simple values directly in memory.

| **Type** | **Size** | **Example** | **Range** |
| --- | --- | --- | --- |
| byte | 1 byte | byte b = 10; | -128 to 127 |
| short | 2 bytes | short s = 1000; | -32,768 to 32,767 |
| int | 4 bytes | int i = 42; | -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | long l = 900L; | ±9 quintillion |
| float | 4 bytes | float f = 3.14f; | ~6-7 decimal digits |
| double | 8 bytes | double d = 3.1415; | ~15-16 decimal digits |
| char | 2 bytes | char c = 'A'; | Unicode 0 to 65,535 |
| boolean | 1 byte\* | boolean b = true; | true or false |

\*Actual size of boolean is JVM-dependent.

**🔸 Non-Primitive Data Types**

These store references to objects.

* String → String name = "Faizan";
* Array → int[] nums = {1, 2, 3};
* Class, Interface, Object → Custom types

**🔑 Java Keywords**

Java has **50+ reserved keywords** that cannot be used as identifiers. Here’s a categorized snapshot:

**🚀 Access Control**

* public, private, protected

**🧱 Class & Object Structure**

* class, interface, extends, implements, new, this, super

**🔁 Control Flow**

* if, else, switch, case, default
* for, while, do, break, continue, return

**⚙️ Modifiers**

* static, final, abstract, synchronized, volatile, transient

**🧠 Exception Handling**

* try, catch, finally, throw, throws

**🧪 Primitive Types**

* int, float, double, char, boolean, byte, short, long

**🧬 Others**

* void, package, import, instanceof, enum, assert

**🧩 Bonus Tip for Interviews**

Use analogies:

* **Primitive types** = raw ingredients.
* **Non-primitive types** = recipes or containers.
* **Keywords** = grammar rules of Java’s language.