# Introduction to Java

Java is a high-level, object-oriented programming language developed by James Gosling at Sun Microsystems (now owned by Oracle) and released in 1995. It follows the Write Once, Run Anywhere (WORA) principle, meaning Java programs can run on any system that has a Java Virtual Machine (JVM).

**Features of Java**

* **Platform-Independent**: Java programs run on any OS with a compatible JVM.
* **Object-Oriented**: Supports principles like encapsulation, inheritance, and polymorphism.
* **Robust and Secure**: Includes strong memory management, exception handling, and security features.
* **Multithreading**: Supports concurrent execution of multiple threads.
* **High Performance**: Uses Just-In-Time (JIT) compilation to improve speed.

## JDK, JRE, and JVM

Java consists of three main components:

### 1. Java Virtual Machine (JVM)

JVM is an abstract machine that provides a runtime environment for Java applications. It converts **bytecode** into machine-specific code. The key tasks of JVM include:

* **Loading** the bytecode
* **Verifying** and **executing** the code
* **Memory management (Garbage Collection)**

### 2. Java Runtime Environment (JRE)

JRE is a part of the **JDK** and includes:

* **JVM**
* **Core libraries**
* **Other supporting files** to run Java applications.

JRE **does not** include development tools like the compiler.

### 3. Java Development Kit (JDK)

JDK is a complete software development package that includes:

* **JRE (JVM + Libraries)**
* **Java Compiler (javac)**
* **Debugger and other development tools**

**JDK = JRE + Development Tools**

### JDK Versions

Java has different versions, such as:

* **JDK SE (Standard Edition)** – For general-purpose programming.
* **JDK EE (Enterprise Edition)** – For large-scale enterprise applications.
* **JDK ME (Micro Edition)** – For mobile and embedded devices.

### Summary Table

| **Component** | **Purpose** |
| --- | --- |
| **JVM** | Converts bytecode to machine code and executes it. |
| **JRE** | Contains JVM and libraries required to run Java programs. |
| **JDK** | Includes JRE + development tools (compiler, debugger, etc.). |

Here’s a concise note on **Basic Syntax: Variables, Data Types, and Operators in Java** 🚀

## Java Basic Syntax

Every Java program starts with a **class**.

The main method is the entry point of the program:

public class HelloWorld {

public static void main(String[] args) {

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

}

}

Java is **case-sensitive** (Hello and hello are different).

Statements end with a **semicolon (;)**.

Blocks of code are enclosed in **curly braces {}**.

## Variables in Java

A **variable** is a container for storing data. It has a **type** and a **name**.

**Declaring Variables:**

dataType variableName = value;

**Example:**

int age = 25;

String name = "Bibek";

double price = 99.99;

**Types of Variables:**

1. **Local Variable** – Declared inside a method, accessible only within it.
2. **Instance Variable (Non-static Variable)** – Belongs to an object, declared inside a class but outside methods.
3. **Static Variable (Class Variable)** – Shared among all objects of a class using static.

## Data Types in Java

Java has **two categories** of data types:

### A. Primitive Data Types (8 types)

| **Data Type** | **Size** | **Example** |
| --- | --- | --- |
| byte | 1 byte | byte b = 100; |
| short | 2 bytes | short s = 32000; |
| int | 4 bytes | int num = 10; |
| long | 8 bytes | long bigNum = 100000L; |
| float | 4 bytes | float pi = 3.14f; |
| double | 8 bytes | double price = 99.99; |
| char | 2 bytes | char letter = 'A'; |
| boolean | 1 bit | boolean isJavaFun = true; |

### B. Non-Primitive Data Types

* **String**: String name = "Java";
* **Arrays**: int[] numbers = {1, 2, 3};
* **Classes & Objects**: Custom-defined data types.

## Operators in Java

Operators perform operations on variables and values.

### A. Arithmetic Operators

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| + | Addition | a + b |
| - | Subtraction | a - b |
| \* | Multiplication | a \* b |
| / | Division | a / b |
| % | Modulus (Remainder) | a % b |

### B. Relational (Comparison) Operators

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| == | Equal to | a == b |
| != | Not equal to | a != b |
| > | Greater than | a > b |
| < | Less than | a < b |
| >= | Greater than or equal to | a >= b |
| <= | Less than or equal to | a <= b |

### C. Logical Operators

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| && | Logical AND | (a > 5 && b < 10) |
| ` |  | ` |
| ! | Logical NOT | !(a > b) |

### D. Assignment Operators

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| = | Assign value | a = 10 |
| += | Add & assign | a += 5 (same as a = a + 5) |
| -= | Subtract & assign | a -= 5 |
| \*= | Multiply & assign | a \*= 5 |
| /= | Divide & assign | a /= 5 |

### E. Unary Operators

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| ++ | Increment | a++ (Post-increment) or ++a (Pre-increment) |
| -- | Decrement | a-- (Post-decrement) or --a (Pre-decrement) |

**Example Program Using Variables, Data Types, and Operators**

public class JavaBasics {

public static void main(String[] args) {

int a = 10, b = 5;

boolean isJavaFun = true;

double sum = a + b;

System.out.println("Sum: " + sum);

System.out.println("Is Java fun? " + isJavaFun);

System.out.println("10 > 5: " + (a > b));

}

}

## Array

An array in Java is a data structure that stores multiple values of the same data type in contiguous memory locations. Arrays have a fixed size and can store both primitive and object types.  
  
### Declaration and Initialization:  
1. \*\*Single Dimensional Array:\*\*  
 ```java  
 int[] numbers = new int[5]; // Declaration and memory allocation  
 int[] values = {10, 20, 30, 40, 50}; // Direct initialization  
 ```  
  
2. \*\*Accessing Array Elements:\*\*  
 ```java  
 numbers[0] = 5; // Assigning value  
 System.out.println(numbers[0]); // Accessing value  
 ```  
  
3. \*\*Iterating Through an Array:\*\*  
 ```java  
 for (int i = 0; i < values.length; i++) {  
 System.out.println(values[i]);  
 }  
 // Using Enhanced For Loop  
 for (int num : values) {  
 System.out.println(num);  
 }  
 ```