## First Java Program

```
public class HelloIshan {
    public static void main(String[] args) {
        System.out.println("Hello Ishan");
    }
}
```

---

## Line-by-Line Explanation:

**public class HelloIshan {**

- **public**: Access modifier, meaning this class can be accessed from anywhere.
- **class**: Keyword to define a class.
- **HelloIshan**: Name of the class (must match the file name: HelloIshan.java).

---

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

- **public**: So Java can call this method from outside the class.
- **static**: So it can run without creating an object of the class.
- **void**: This method does not return any value.
- **main**: Entry point of any Java application.
- **String[] args**: Accepts command-line arguments as an array of Strings.

---

**System.out.println("Hello Ishan");**

- **System**: A built-in Java class that provides access to system resources.
- **out**: An object of PrintStream connected to the console.
- **println()**: A method to print a message and move to the next line.
- **"Hello Ishan"**: The message being printed.

▶️ **Output:**

Hello Ishan

**1. Features of Java – Bytecode, JVM, and JDK**

**a. Bytecode**

- Java source code (.java) is **compiled** into an intermediate form called **Bytecode** (.class file).
- Bytecode is **platform-independent**, meaning it can run on any system that has a **Java Virtual Machine (JVM)**.
- Makes Java a **"write once, run anywhere"** language.

**b. Java Virtual Machine (JVM)**

- JVM is a **runtime environment** that executes Java bytecode.
- Each OS has its own version of JVM.
- JVM performs:
    - **Loading** of class files
    - **Verifying** bytecode
    - **Executing** code
    - **Memory Management** (Garbage Collection)

**c. Java Development Kit (JDK)**

- JDK is a complete software package to develop Java applications.
- Includes:
    - **Java Compiler (javac)**
    - **Java Runtime Environment (JRE)**
    - **JVM**
    - Development tools (e.g., javap, javadoc, debugger)

**2. Data Types in Java**

Java is a **statically typed** language — data types must be declared before use.

**a. Primitive Data Types**

| Type | Description | Example |
|---|---|---|
| int | Integer numbers | int age = 25; |

| Type | Description | Example |
|---|---|---|
| float | Floating point numbers (single-precision) | float pi = 3.14f; |
| double | Floating point (double-precision) | double d = 99.99; |
| char | Single character | char grade = 'A'; |
| boolean | True or false | boolean flag = true; |

**b. Type Conversion**

- **Implicit Conversion (Widening)**:
    - Small data type is converted to a larger type automatically.
    - Example:
    - int x = 10;
    - double y = x;  // int to double

**c. Type Casting (Explicit Conversion)**

- Convert larger type to smaller type **manually**.
- May lead to data loss.
- Syntax: (targetType) value
- Example:
- double d = 10.5;
- int i = (int) d;  // i becomes 10

**Summary:**

- **Bytecode + JVM** make Java platform-independent.
- **JDK** is required to develop and run Java apps.
- Java supports various **primitive data types** with clear memory allocation.
- **Type conversion** can be implicit or explicit, depending on direction and size.

# Operators

---

✅ **Java Operators – Summary Notes**

🔹 **1. Arithmetic Operators**

Used for basic math operations:

- +, -, *, /, %
  Example: a + b, a % b

---

🔹 **2. Bitwise Operators**

Operate on bits:

- &, |, ^, ~, <<, >>
  Example: a & b, a << 1

---

🔹 **3. Relational Operators**

Compare two values:

- ==, !=, >, <, >=, <=
  Returns true or false.

---

🔹 **4. Logical Operators**

Work with boolean values:

- && (AND), || (OR), ! (NOT)
  Used in conditions.

---

🔹 **5. Assignment Operators**

Assign values and combine with operations:

- =, +=, -=, *=, /=, %=
  Example: x += 5 → x = x + 5

---

🔹 **6. Operator Precedence**

int result = 10 + 5 * 2 - 4 / 2;

**Evaluation steps:**

1. 5 * 2 = 10
2. 4 / 2 = 2
3. 10 + 10 - 2 = 18

**Final Result:** 18