5th March Task

Source Code: This is the human-readable Java code written by developers in .java files.

Ex:

public class HelloWorld {

public static void main(String[] args) {

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

}

}

Byte Code: After compilation using the **Java Compiler (javac)**, the .java file is converted into **bytecode**, which is stored in a .class file.

Bytecode is a **platform-independent** intermediate representation that can be executed on any system with a Java Virtual Machine (JVM).

Example (Bytecode representation in .class file, viewed using javap -c HelloWorld.class):

0: getstatic #7 // Field java/lang/System.out:Ljava/io/PrintStream;

3: ldc #9 // String Hello, World!

5: invokevirtual #11 // Method java/io/PrintStream.println:(Ljava/lang/String;)V

8: return

Native Code: When the JVM executes the **bytecode**, it converts it into **native machine code** (specific to the operating system and processor).

Example: The final instructions executed at the processor level will be in **binary/machine code**, which is not human-readable.

**Execution Flow in Java**

1. **Write** Java source code (.java file).
2. **Compile** using javac → Generates **bytecode** (.class file).
3. **Run** using java command → JVM **interprets** or **compiles** it into **native code** using JIT.
4. **Execute** on the hardware as machine-specific **native instructions**.

Why is Java both a Compiled and Interpreted Language?

Java is both **compiled and interpreted** because it follows a two-step execution process. First, the **Java Compiler (javac) compiles** the source code into **platform-independent bytecode**. Then, the **JVM interprets** the bytecode and converts it into **native machine code** using the **JIT compiler**, enabling Java to run on any system with a JVM.