03-How Java Works

When we write a Java program, we typically use tools like JShell to test our code. However, running a complete program requires understanding the role of the Java Virtual Machine (JVM) and the Java Runtime Environment (JRE). Let's delve into how JVM operates and the control flow of a Java program.

The Role of JVM

The JVM is an integral part of the Java ecosystem, allowing Java code to be executed on any platform. Each operating system (OS) has a specific JVM designed for it, making Java platform-independent at the source code level but platform-dependent at the binary level.

Diagram

Layers

Control Flow of Java Code

Java code undergoes several stages from writing to execution. Here's a step-by-step breakdown of the process:

1. **Writing Code**: Java code is written in simple, understandable English terms.
2. **Compilation**: The Java compiler converts the source code (.java files) into bytecode (.class files).
3. **Execution by JVM**: The JVM executes the bytecode. It specifically looks for the main method as the entry point for any Java program.

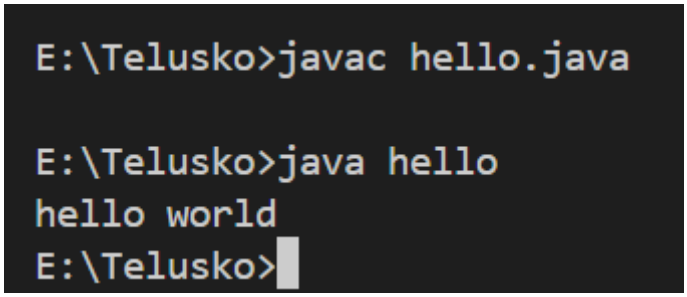
Main Method:

- The main method has a specific signature: public static void main (String[] args) {}.
- This method is crucial as it marks the starting point of program execution.

Steps to Write and Execute Java Code

1. **Write the Code**: Ensure all executable code is within the main method.
2. **Follow OOP Principles**: Java is object-oriented; thus, everything is treated as an object, and classes act as blueprints.
3. **Compile the Code**: Use the Java compiler to compile your code.
    - This generates a .class file containing bytecode.
4. **Run the Code**: Use the java command followed by the class name (without the .class extension) to run your program.

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



Understanding JRE

The JRE is essential for running Java programs. It includes the JVM and a set of libraries and other files that the JVM uses at runtime. Think of JRE as a kitchen:

- **Kitchen Analogy:**
    - The kitchen provides the environment to cook a dish.
    - Utensils and Ingredients are like the libraries and resources JRE provides.

Example:

- Suppose you want to bake a cake. The JRE is like the entire kitchen setup needed for baking, including the oven, mixing bowls, and ingredients. The JVM is the oven that actually bakes the cake using the ingredients (bytecode) provided.

Visualizing Java Architecture

Imagine the Java development environment as nested boxes:

- **Outer Box (JDK)**: The Java Development Kit includes everything for development.
- **Middle Box (JRE)**: The JRE provides the runtime environment.

- **Innermost Box (JVM)**: The JVM interprets and executes the bytecode.