### **Visual Overview**

### 

### 

### 

### 

### 

### 

### **Java Compilation Process**

1. **Writing the Code:**
   * A programmer writes the Java code in a .java file.
   * This file contains source code written in the Java programming language.
2. **Compilation:**
   * The Java source code is then compiled by the Java Compiler (javac).
   * The compiler checks the code for syntax errors and, if there are no errors, translates the Java source code into bytecode.
   * Bytecode is a set of instructions that looks like machine code but is not specific to any processor.
   * The bytecode is stored in a .class file.
3. **Loading the Bytecode:**
   * The bytecode is loaded into the Java Virtual Machine (JVM) by the class loader.
   * The class loader loads the .class file into the JVM.
4. **Bytecode Verification:**
   * Before the bytecode is executed, the JVM verifies it to ensure that it is valid and does not violate any security constraints. This is done by the bytecode verifier.
5. **Execution:**
   * Finally, the JVM executes the bytecode. The JVM translates the bytecode into machine code that the underlying operating system can understand and execute.
   * This is done using the Just-In-Time (JIT) compiler, which converts bytecode into native machine code at runtime.

### 

### 

### 

### 

### 

### 

### 

### 

### **Steps in Detail:**

1. **Writing the Code:**

Example: **HelloWorld.java**  
public class HelloWorld {

public static void main(String[] args) {

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

}

}

1. **Compilation:**
   * Command: javac HelloWorld.java
   * This command invokes the Java compiler to compile HelloWorld.java and generate HelloWorld.class.
2. **Loading the Bytecode:**
   * The class loader loads HelloWorld.class into the JVM.
3. **Bytecode Verification:**
   * The JVM verifies the bytecode to ensure it is correct and secure.
4. **Execution:**
   * Command: java HelloWorld
   * This command starts the JVM, which then executes the main method of the HelloWorld class. The System.out.println("Hello, World!"); line prints "Hello, World!" to the console.

### 

### 

### 

### 

### 

### 

### 

### 

### **Example Code Explanation**

Let's look at the example code in detail:

public class HelloWorld {

public static void main(String[] args) {

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

}

}

* public class HelloWorld { }: This defines a public class named HelloWorld. In Java, every application begins with a class definition.
* public static void main(String[] args) { }: This is the main method, which is the entry point for any Java application. The main method is always public, static, and void, and it takes a String array as an argument.
* System.out.println("Hello, World!");: This line prints "Hello, World!" to the console. System.out is a standard output stream, and println is a method that prints the argument passed to it, followed by a new line.

### 

### 

### **Summary**

1. **Write Java Code**: Create a .java file with your Java code.
2. **Compile Java Code**: Use the javac compiler to translate the Java code into bytecode, resulting in a .class file.
3. **Load Bytecode**: The JVM class loader loads the .class file.
4. **Verify Bytecode**: The JVM verifies the bytecode for correctness and security.
5. **Execute Bytecode**: The JVM executes the bytecode using the JIT compiler, translating it to machine code and running it on your computer.

Prepared by : Rishi Patel