Static block

The Java Virtual Machine (JVM) executes Java bytecode, which is generated from Java source code. When a class is loaded by the JVM, it goes through a series of steps, including the execution of static blocks.

Static blocks in Java are used to initialize static variables or perform any other one-time initialization tasks for a class. They are executed when the class is loaded by the JVM, before any other code in the class is executed.

Here is the working process of JVM for static blocks:

1. Loading: The JVM loads the bytecode of a class into memory when it is first referenced in the program. This process involves locating the class file, reading its bytecode, and creating a representation of the class in memory.

2. Linking: After loading, the JVM performs the linking phase, which consists of three steps: verification, preparation, and resolution.

- Verification: The JVM verifies the bytecode to ensure it is valid and doesn't violate any security constraints. It checks for bytecode integrity, type safety, and other rules defined by the Java language specification.

- Preparation: In this step, the JVM allocates memory for static variables and initializes them with their default values (zero for numeric types, false for boolean, null for object references).

- Resolution: The JVM resolves symbolic references in the bytecode to direct references. This includes resolving references to other classes, methods, and fields.

3. Initialization: Once the linking phase is complete, the JVM proceeds to initialize the class. This involves executing the static blocks and initializing static variables in the order they appear in the code.

- Static blocks: Static blocks are executed in the order they appear in the class. They can contain any valid Java code and are typically used for static variable initialization or other one-time setup tasks.

- Static variables: After executing the static blocks, the JVM initializes static variables with the values assigned in the code.

Static blocks are executed only once, when the class is loaded by the JVM. They are useful for performing initialization tasks that need to be done before any instances of the class are created or any methods are called.

Here's an example of a class with a static block:

public class MyClass {

static {

// Static block code

System.out.println("Static block executed");

}

public static void main(String[] args) {

// Main method code

System.out.println("Main method executed");

}

}

Output:

Static block executed

Main method executed