Lifecycle of a Program in Java

The **Lifecycle of a Program in Java** refers to the different stages a Java program goes through from **writing the code** to **execution and termination**. It can be described as follows:

1. Writing the Program (Source Code Stage)

- The program is written in a `.java` file using an editor or IDE.
- Example:

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

2. Compilation Stage

- The Java compiler (`javac`) converts the `.java` file into **bytecode** (`.class` file).
- Bytecode is platform-independent.
- Example command:

```
javac Hello.java
```

This produces `Hello.class`.

3. Class Loading Stage

- The **ClassLoader** loads the `.class` file into JVM memory.
- It links it with required libraries and system classes.

4. Bytecode Verification

- The JVM's **Bytecode Verifier** checks the bytecode for:
  - Security issues
  - Memory violations
  - Stack overflows or illegal access
- Ensures the code is safe before execution.

5. Execution Stage (Runtime)

- The **JVM Interpreter** or **JIT (Just-In-Time) Compiler** translates bytecode into native machine code.
- The program runs and produces output.
- Example output:

```
Hello, World!
```

6. Program Termination

- Once the `main()` method completes, the program ends.
- The **Garbage Collector (GC)** frees up unused memory.
- The JVM shuts down.

Summary

1. Writing → `.java` file
2. Compilation → `.class` (bytecode)
3. Class Loading → Load into JVM
4. Verification → Safe check
5. Execution → Interpreter/JIT run code
6. Termination → Memory cleanup + exit