**Features of Java**

* Simple and Easy to Learn: Java is designed to be easy to read and write.
* Object-Oriented Programming: Everything in Java is treated as an object.
* Platform Independence: Java programs can run on any device with the JVM.
* Automatic Memory Management: Java handles memory allocation and deallocation automatically.
* Security: Java has built-in security features to protect against threats.
* Rich API: Java provides a wide range of standard APIs for various tasks.
* Multithreading: Java supports multiple threads of execution for better performance.
* High Performance: Java is optimized for performance through Just-In-Time compilation and other techniques.

**Internal Working of Java:**

* Compilation:

Write Java code in a .java file.

Java Compiler (javac) converts it to bytecode (.class file).

* Bytecode:

Platform-independent intermediate code.

* Execution:

The JVM interprets the bytecode into machine code specific to the hardware.

JDK/JRE/JVM:

**JVM:**

Converts bytecode into machine code.

Contains an interpreter/JIT compiler.

Different JVMs for different platforms.

**JRE:**

Includes the JVM and standard class libraries.

Needed to run Java programs.

**JDK:**

Includes tools for developing Java programs (compiler, debugger).

Contains JRE to run the programs.

**Difference Between Bytecode and Machine Code:**

|  |  |
| --- | --- |
| Byte Code | Machine Code |
| It is platform independent code | It is platform dependent code |
| It is understand by JVM | It is understand by OS |
| It is larger in size | It is smaller in size |
| More secure then machine code | Less secure then byte code |

**\*Internal Details of a Java Program**

**Compile Time:**

Source Code (HelloWorld.java) -> [Java Compiler] -> Bytecode (HelloWorld.class)

**Runtime:**

Classloader: Loads class files into memory.

Bytecode Verifier: Ensures no illegal code.

Interpreter: Executes bytecode instructions.

**Flow:**

Write Code -> Compile Code -> Load Class -> Verify Bytecode -> Execute Code