**Advantages of Java:**

**Simple:** Java is much simpler than C++ because Java uses automatic memory allocation and garbage collection.

**Object-Oriented:**It is complete object oriented.

**Platform-Independent:** Ability to move easily from one computer system to another

**Secure:** The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.

**Allocation:** Java has the feature of Stack allocation system. It helps the data to be stored and can be restored easily.

**Multithreaded:** The capability for a program to perform several tasks simultaneously within a program.

**Disadvantages of Java:**

**Performance:** SIgnificantly slower and more memory-consuming than C or C++.

**Look and feel:** The default look and feel of GUI applications written in Java using the Swing toolkit is very different from native applications.

**Single-paradigm language:** The addition of static imports in Java 5.0 the procedural paradigm is better accommodated than in earlier versions of Java.

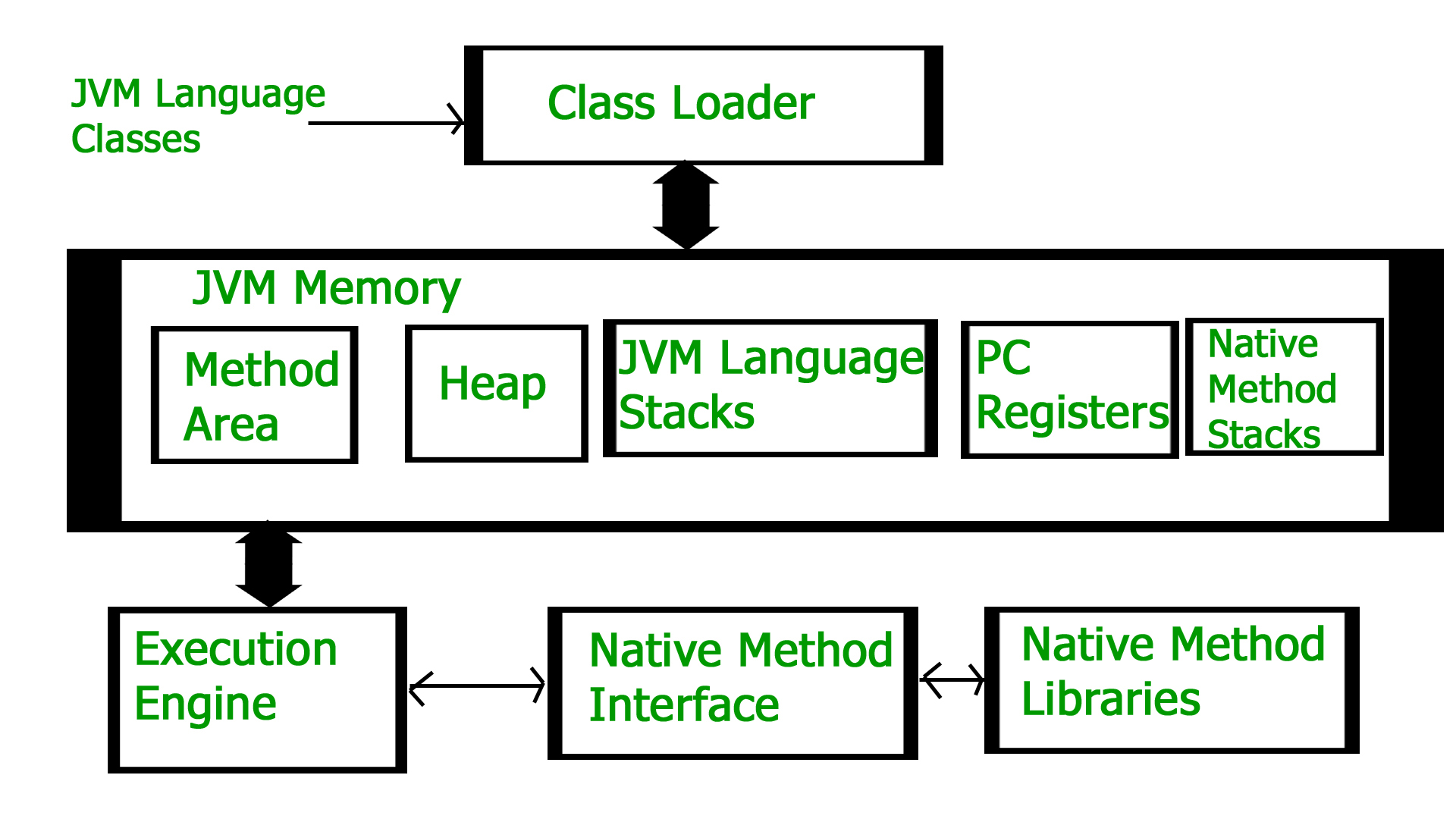
**Multitasking and Multithreading:**

1. The basic difference between multitasking and multithreading is that in multitasking, the system allows executing multiple programs and tasks at the same time, whereas, in multithreading, the system executes multiple threads of the same or different processes at the same time.
2. Multitasking allocatesseparatememory and resources for each process/program whereas, in multithreading threads belonging to the same processshares the same memory and resources as that of the process.

Multitasking is similar to multiprogramming whereas, Multithreading is thread-based multitasking. Multithreading is less costlier than multitaskings as threads are easy to create then a process.

|  |  |  |
| --- | --- | --- |
|  | **MULTIPROCESSING** | **MULTITHREADING** |
| Basic | Multiprocessing adds CPUs to increase computing power. | Multithreading creates multiple threads of a single process to increase computing power. |
| Execution | Multiple processes are executed concurrently. | Multiple threads of a single process are executed concurrently. |
| Creation | Creation of a process is time-consuming and resource intensive. | Creation of a thread is economical in both sense time and resource. |
| Classification | Multiprocessing can be symmetric or asymmetric. | Multithreading is not classified. |

**Java Virtual Machine(JVM):**



A VirtualMachine is a software implementation of a physical machine. JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is a part of JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java enabled system without any adjustment. This is all possible because of JVM.

JVM is divided into three main subsystems:

1. Class Loader Subsystem
2. Runtime Data Area
3. Execution Engine

**1. Class loader subsystem:** JVM's class loader subsystem performs 3 tasks

a. It loads .class file into memory.

b. It verifies byte code instructions.

c. It allots memory required for the program.

**2. Run time data area:** This is the memory resource used by JVM and it is divided into 5 parts

**a.** Method area: Method area stores class code and method code.

**b.** Heap: Objects are created on heap.

**c.** Java stacks**:** Java stacks are the places where the Java methods are executed. A Java stack contains frames. On each frame, a separate method is executed.

**d.** Program counter registers: The program counter registers store memory address of the instruction to be executed by the microprocessor.

**e**. Native method stacks: The native method stacks are places where native methods (for example, C language programs) are executed. Native method is a function, which is written in another language other than Java.

**3. Native method interface:** Native method interface is a program that connects native methods libraries (C header files) with JVM for executing native methods.\

**4. Native method library:** holds the native libraries information.

**5. Execution engine:** Execution engine contains interpreter and JIT compiler, which covert byte code into machine code. JVM uses optimization technique to decide which part to be interpreted and which part to be used with JIT compiler. The HotSpot represent the block of code executed by JIT compiler.

**JDK vs JRE vs JVM:**

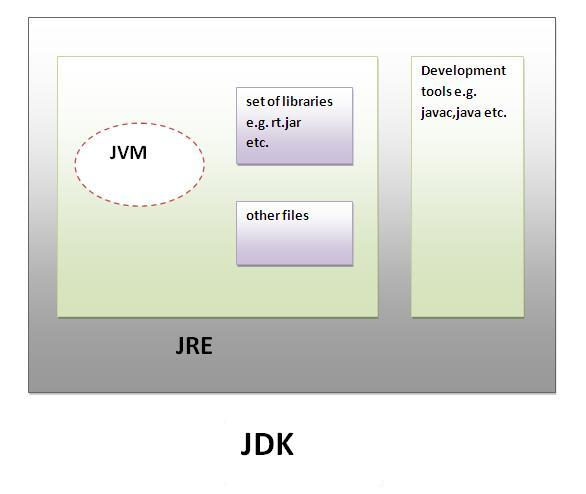
The **JRE** is the **Java Runtime Environment**. It is a package of everything necessary to run a compiled Java program, including the Java Virtual Machine (JVM), the Java Class Library, the javacommand, and other infrastructure. However, it cannot be used to create new programs.

The **JDK** is the **Java Development Kit**, the full-featured SDK for Java. It has everything the JRE has, but also the compiler (javac) and tools (like javadoc and jdb). It is capable of creating and compiling programs

**Note :** JDK is only used by Java Developers.

JRE = JVM + libraries to run java application

JDK=JRE + tools to develop java application



**Drawbacks of C and C++:**

1.) It has no security

2.) Complex in a very large high-level program.

3.) Used for platform specific application commonly.

4.) For a particular operating system or platform, the library set has usually chosen that locks.

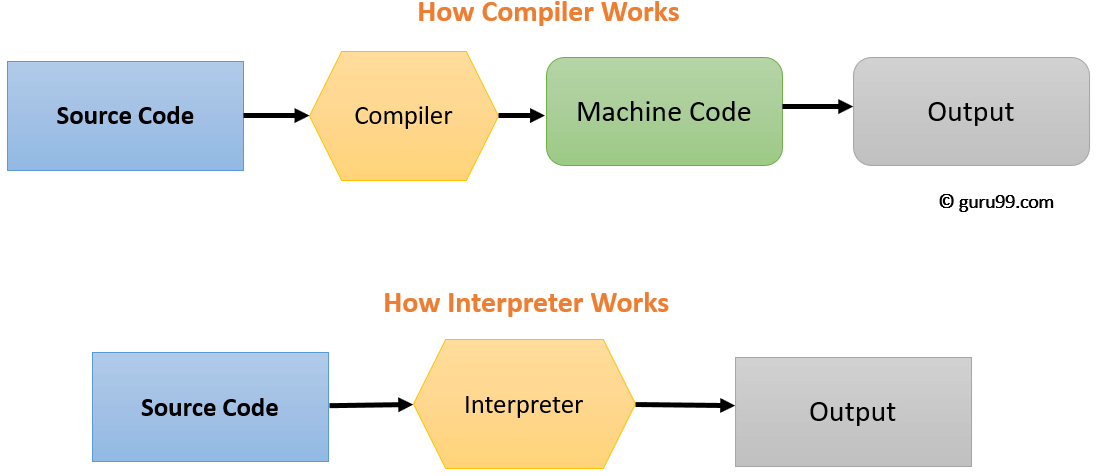
5.) When C++ used for web applications complex and difficult to debug.

6.) C++ can’t support garbage collection.

7.) C++ is not secure because it has a pointer, friend function, and global variable.

8.) No support for threads built in.

**Compiler vs Interpreter:**



|  |  |
| --- | --- |
| Interpreter | Compiler |
| Translates program one statement at a time. | Scans the entire program and translates it as a whole into machine code. |
| It takes less amount of time to analyze the source code but the overall execution time is slower. | It takes large amount of time to analyze the source code but the overall execution time is comparatively faster. |
| No intermediate object code is generated, hence are memory efficient. | Generates intermediate object code which further requires linking, hence requires more memory. |
| Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy. | It generates the error message only after scanning the whole program. Hence debugging is comparatively hard. |
| Programming language like Python, Ruby use interpreters. | Programming language like C, C++ use compilers. |