**Features of Java**

These are also known as java buzzwords.

Some are listed below:

1.Simple

2.Object Oriented

3.Platform independent

4.Secured

5.Robust

6.Multithreaded

7.Distributed

8.Dynamic

**1.Simple**: If we know the C or C++ it is easy to learn. Its syntax is simple. There are no pointers and it contains an automatic garbage collector.

**2. Object-Oriented:** Java contains all oop concepts like Class, Object, Inheritance, Polymorphism, Abstraction, Encapsulation.

**3.Platform independent:** We can run Java code in any type of platform.

**4. Secured:** There are no explicit pointers in Java and Java programs are run inside the virtual machine. It contains Classloader, Bytecode verifier, Security manager.

**5.Robust:** Java uses strong memory management, exception handling, there is a lack of pointers that avoids security problems.

**6.Multithreading:** A thread is like a separate program, executing concurrently. The main advantage of the multithreading is that it doesn’t occupy memory for each thread.

**7. Distributed:** User can create Distributed applications using Java.

**8.Dynamic:** Java supports dynamic loading of classes.

**Drawbacks of C and C++ over Java**

\*C and C++ are platform dependent.

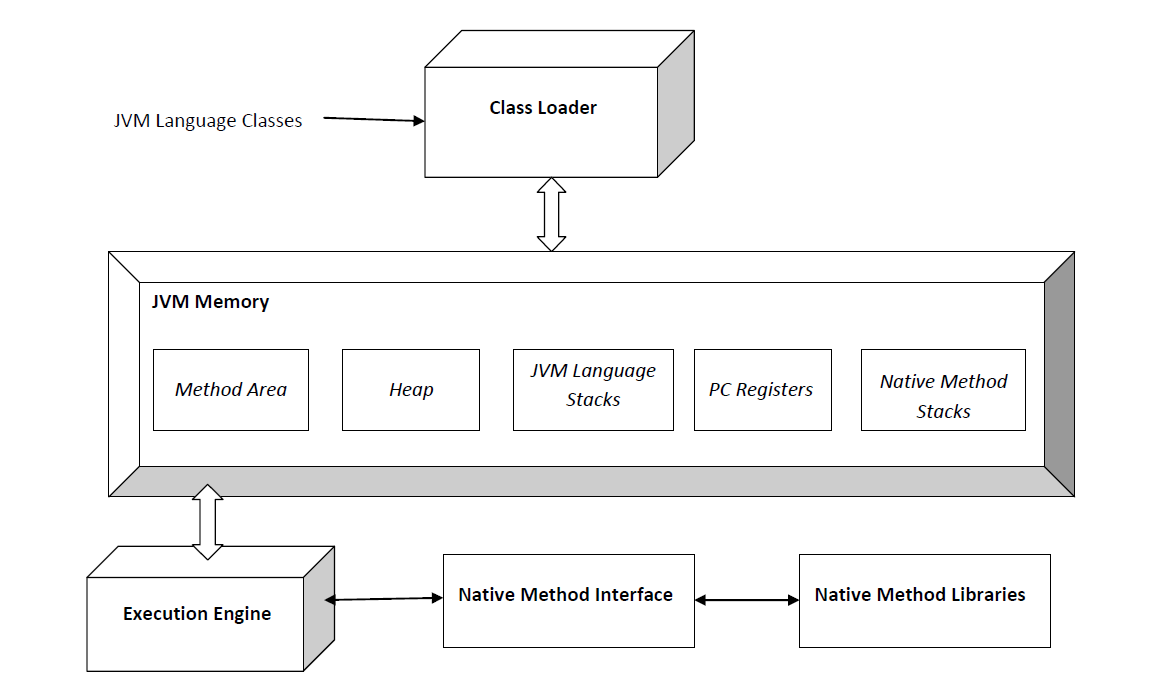
\*C and C++ are don’t have an automatic garbage collector.

\*C and C++ are not secured like Java.

\*C is not an object-oriented programming language and C++ is not pure object-oriented

programming language.

**JVM**



**Classloader:**

Classloader performs Loading, Linking, Initialization.

**1. Loading:** finds and imports the binary data for a type.

**2. Linking:** performs verification, preparation, resolution.

**\*Verification:** ensures the correctness of the imported type.

**\*Preparation:** allocates memory for class variables and initializing the memory to

default values

\***Resolution:** transforms symbolic references from the type into direct references.

**3.Initialization:** invokes Java code that initializes class variables to their proper starting values.

In general, there are two types of class loader: bootstrap class loader and user-defined class loader.

**JVM memory:**

**Memory area:** it stores field and method data, the code for methods.

**Heap:** it is the runtime data area in which objects are allocated.

**Stack:** stores local variables, partial results, frames and plays a part in method invocation and return.

**Register:** register contains the address of the Java virtual machine instruction currently executed.

**Native method stack:** it contains all the native methods used in the application.

**Java bin files**

<https://docs.oracle.com/javase/8/docs/technotes/tools/windows/toc.html>

**Java:** To launch a Java application.

**Javap:** The javap command disassembles one or more class files. The output depends on the options used. When no options are used, then the javap command prints the package, protected and public fields, and methods of the classes passed to it. The javap command prints its output to stdout.  
 **Jdb(java debugger):** Finds and fixes bugs in Java platform programs.

**Javac:** Reads Java class and interface definitions and compiles them into bytecode and class files.  
  
**Javadoc:** Generates HTML pages of API documentation from Java source files.  
  
**Jar:** Manipulates Java Archive (JAR) files.

**Javah:** Generates C header and source files from a Java class.