

**ULISES ISAAC ROCHA HUERTA**

**JAVA VIRTUAL MACHINE**

**JAVA VIRTUAL MACHINE**

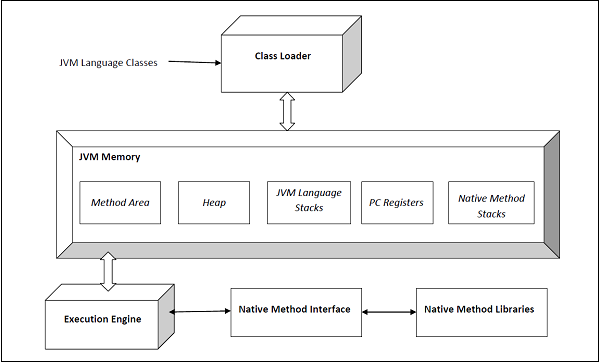
**INTRODUCTION:**

The Java virtual machine is like a computer that has its own data processing memory. The JVM is an engine that manages the system's memory and controls applications at a runtime. It can be run on the host's operating system and its resources are established.

For example, Oracle has its own JVM implementation called HotSpot JVM

IBM has the Oracle JVM Specs source implementation

**DESARROLLO:**

Next, to give an example of the structure of a JVM, I present a diagram of the architecture of the Oracle HotSpot

We use the example of the Oracle JVM. The execution engine contains a garbage collector and the JIT compiler. There are two types of JVM, client-server, and the two types of JVM share the same execution code. The user can choose and control which type he will use, he has to specify the indicators - client or -server.

Another interesting fact is that the JVM server has been designed for long-term JAVA applications on servers.

The JVM has two versions of 32b and 64b and the user can specify the version he will use by typing -d32 or -d64

The 32b version can only address 4G of memory

The 64b version is more advisable to maintain large data sets.

The JVM manages:

1. The loading process
2. Link process
3. Class initialization process
4. Interface initialization process

In the linking process, the loaded classes are combined in the JVM runtime state so that they can be executed during the initialization phase.

CONCLUSION:

It is interesting how the java virtual machine works, it has many processes of management and administration of the system resources created.

It is also important to keep in mind that it is not only for the clients.

It is able to interpret and execute instructions expressed in a special binary code (the Java byte code), which is generated by the Java language compiler