Simple Program of Java

```
1.      class Simple{
2.          public static void main(String args[]){
3.           System.out.println("Hello Java");
4.          }
5.      }
```

save this file as Simple.java

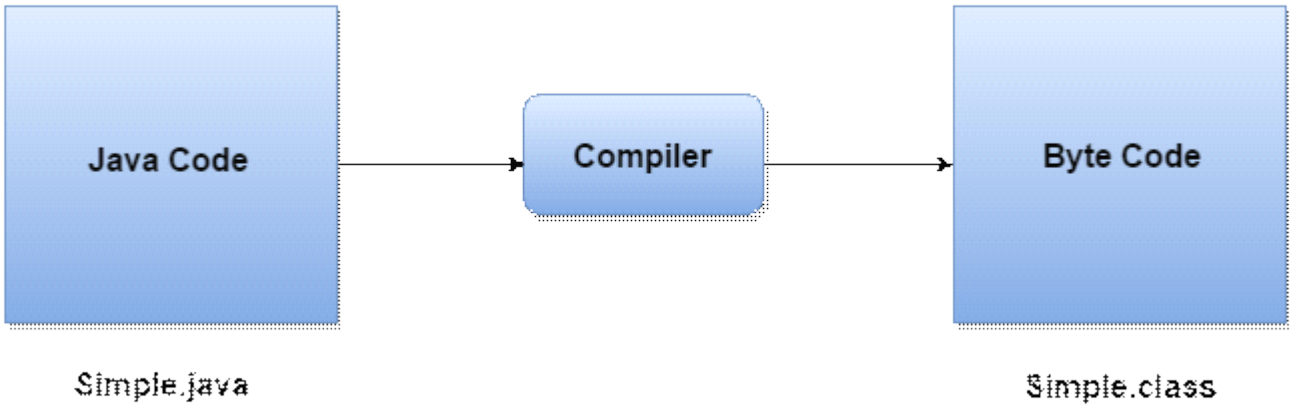
Understanding first java program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

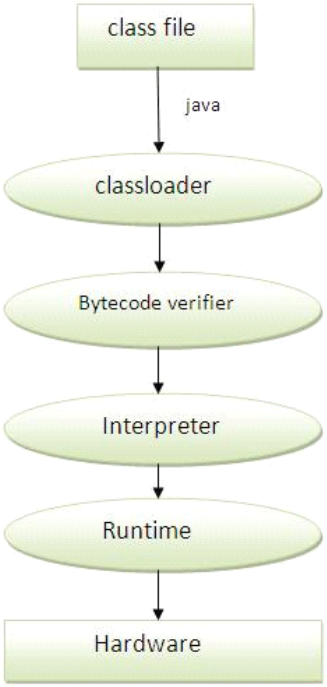
What happens at compile time?

At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.



What happens at runtime?

At runtime, following steps are performed:



Classloader: is the subsystem of JVM that is used to load class files.

Bytecode Verifier: checks the code fragments for illegal code that can violate access right to objects.

Interpreter: read bytecode stream then execute the instructions.

Difference between JDK, JRE and JVM

JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent.

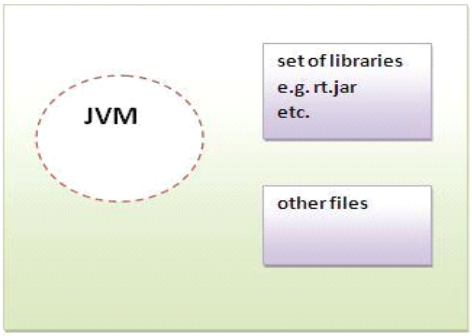
The JVM performs following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JRE

JRE is an acronym for Java Runtime Environment.It is used to provide runtime environment.It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

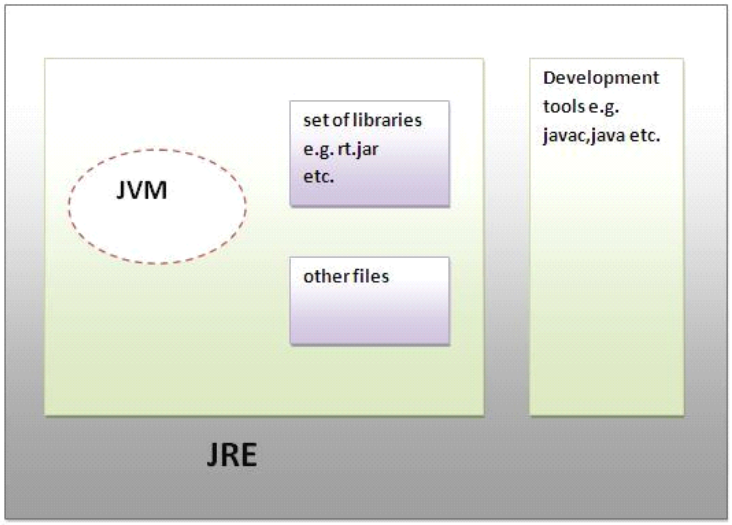
Implementation of JVMs are also actively released by other companies besides Sun Micro Systems.



JRE

JDK

JDK is an acronym for Java Development Kit.It physically exists.It contains JRE + development tools.



JDK

JVM (Java Virtual Machine)

1. Java Virtual Machine
2. Internal Architecture of JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

What is JVM

It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

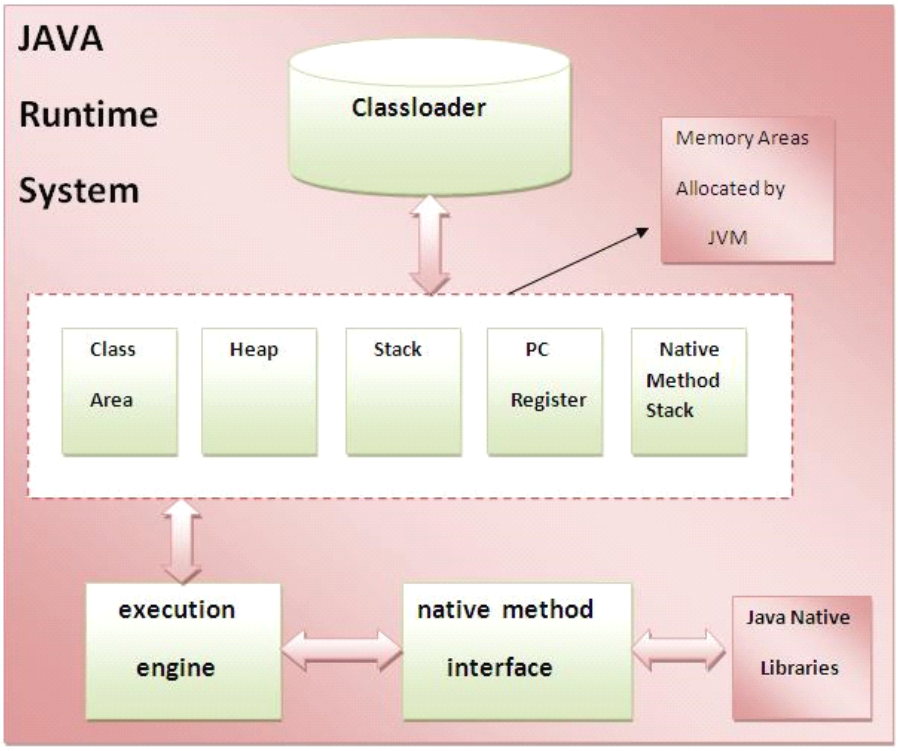
JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap

- Fatal error reporting etc.

Internal Architecture of JVM

Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



1) Classloader

Classloader is a subsystem of JVM that is used to load class files.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames.It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1) A virtual processor

2) Interpreter: Read bytecode stream then execute the instructions.

3) Just-In-Time(JIT) compiler: It is used to improve the performance.JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

Java's Magic: The Byte Code

The means that allows Java to solve both the security and the portability problems is that the output of a Java compiler is not executable code but it is the Bytecode.

Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the *Java Virtual Machine (JVM).*

JVM is an *interpreter for bytecode.* The fact that a Java program is executed by JVM helps solve the major problems associated with downloading programs over the Internet. Translating a Java program into bytecode helps makes it much easier to run a program in a wide variety of environments. This is because only the JVM needs to be implemented for each platform.

Once the run-time package exists for a given system, any Java program can run on it. If a Java program is compiled to its subject code, then different versions of the same program would exist for each type of CPU connected to the Internet. This is not a practical solution. Thus, the understanding of bytecode is the most effective way to create strictly portable programs. Now, if a Java program is interpreted, it also helps to make it secure. Since the execution of every Java program is under the control of the JVM, the JVM can contain the program, and prevent it from generating side effects outside of the system. The use of byte Code enables the Java run-time system to execute programs much faster.

Sun provides a facility called *Just In Time (JIT) compiler* for bytecode. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. JIT compiles the code 3.S and when needed.