About the Java Technology

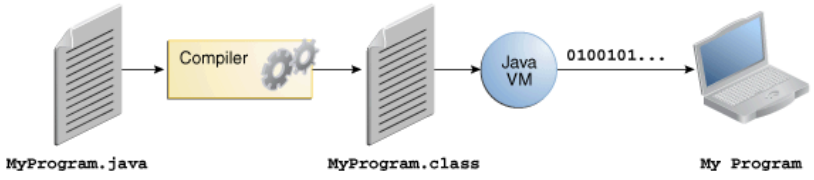
The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High performance
- Robust
- Secure

Java Application Development Architecture

- all source code is first written in plain text files ending with the `.java` extension.
- Those source files are then compiled into `.class` files by the `javac` compiler.
- A `.class` file does not contain code that is native to your processor; it instead contains *bytecodes* — the machine language of the Java Virtual Machine[1] (Java VM)
- The `java` launcher tool then runs your application with an instance of the Java Virtual Machine.

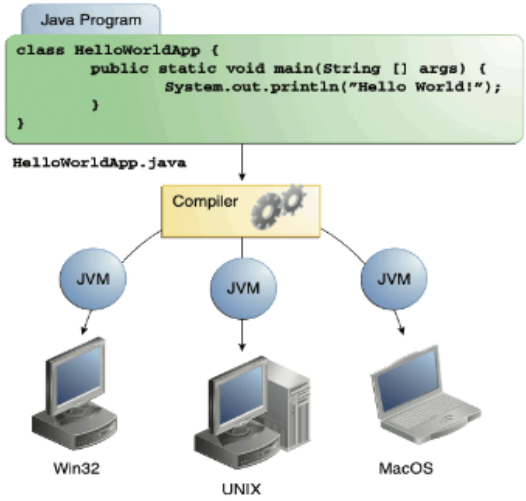


An overview of the software development process.

- Because the Java VM is available on many different operating systems, the same `.class` files are capable of running
- Some virtual machines, such as the Java SE HotSpot at a Glance, perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.

The Java Platform

- A *platform* is the hardware or software environment in which a program runs.
- Most platforms can be described as a combination of the operating system and underlying hardware.
- The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.



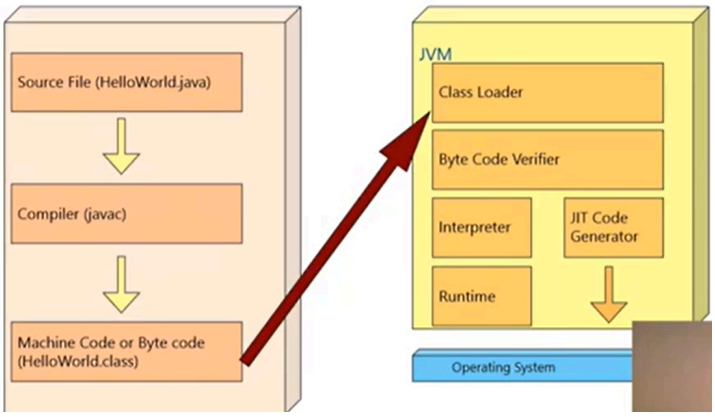
Through the Java VM, the same application is capable of running on multiple platforms.

The Java Platform (JRE) has 2 Components :

- **Java Virtual Machine (JVM)**
- **Java Application Programming Interface (API)**

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as *packages*.

JVM Architecture



Class loader - Loads file HelloWorld.class to RAM.
ByteCode Verifier - Verifies bytecode is correct or not.
Method Area : **Compiled code, final variables, static variables** are kept in the method area.
Heap : All objects are created in heap.

a) **Young generation** : Objects which are newly created
b) **Old generation** : Objects which are old
c) **Permanent space** : All objects required for JVM forever are stored here.

Garbage collectors check these areas and remove objects with no reference.
Stack : All local variables are stored in the stack.
PC : stores address of next instruction to be executed mentioned in bytecode.
Native Stack : Used for calling the methods that are written in other languages.
JIT (Just In time Compiler) / Interpreter : Executes code line by line, jit to check whether executed code is frequently executed or stored in cache or not.
Libraries

JRE = JVM+Libraries

What Can Java Technology Do?

The general-purpose, high-level Java programming language is a powerful software platform. Every full implementation of the Java platform gives you the following features:

- **Development Tools**: The development tools provide everything you'll need for compiling, running, monitoring, debugging, and documenting your applications.
- **Application Programming Interface (API)**: The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access, and more.
- **Deployment Technologies**: The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying your applications to end users.
- **User Interface Toolkits**: The JavaFX, Swing, and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).
- **Integration Libraries**: Integration libraries such as the Java IDL API, JDBC API, Java Naming and Directory Interface (JNDI) API, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.

Overview Of Basic Program :

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

The Hello World program under a microscope:

```
public static void main(String[] args)
```

- the keyword `public` indicates that the method can be invoked from everywhere;
- the keyword `static` indicates the method can be invoked without creating an instance of the class;
- the keyword `void` indicates the method doesn't return any value;
- the array variable `args` contains arguments entered at the command line, the array is empty if there are no arguments.