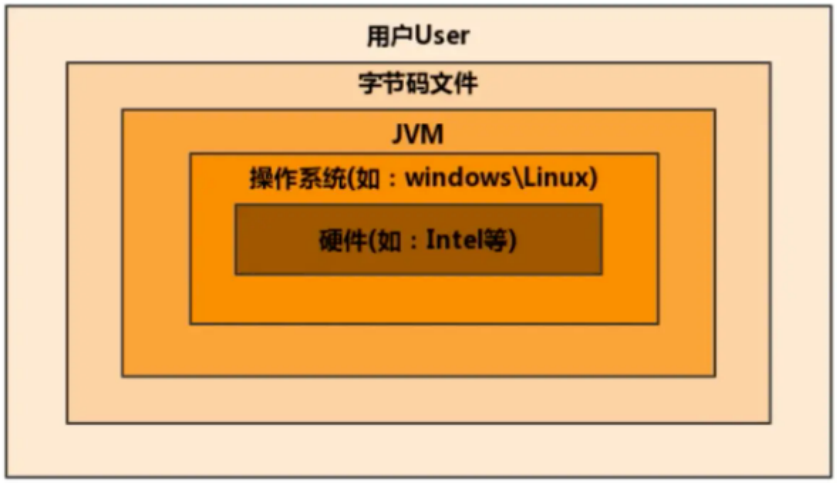
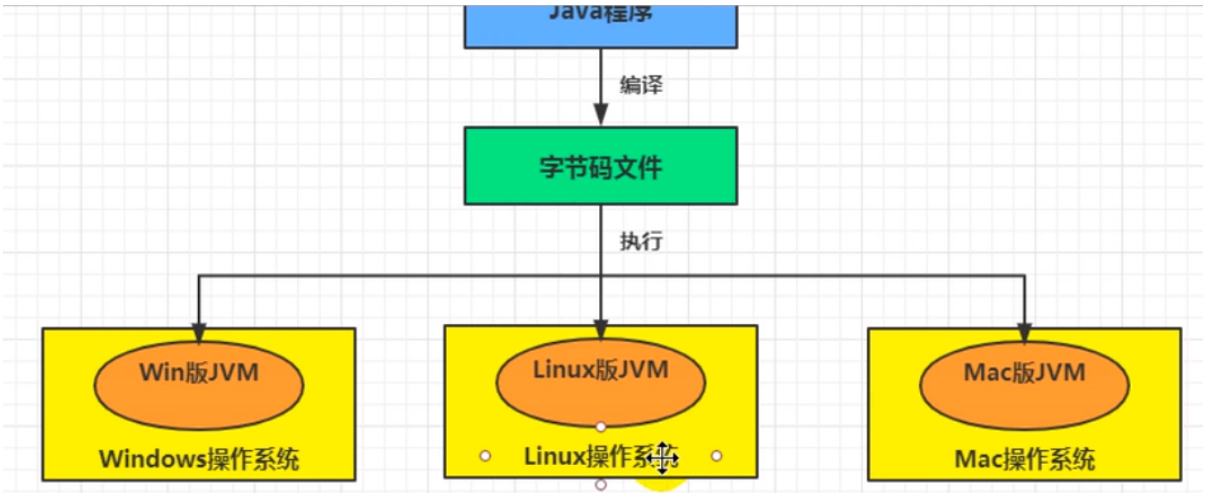
首先来看计算机系统当中JVM所处的位置



JVM是运行在操作系统之上的，并没有和硬件有直接的交互

Java一次编译,到处运行



古今JVM

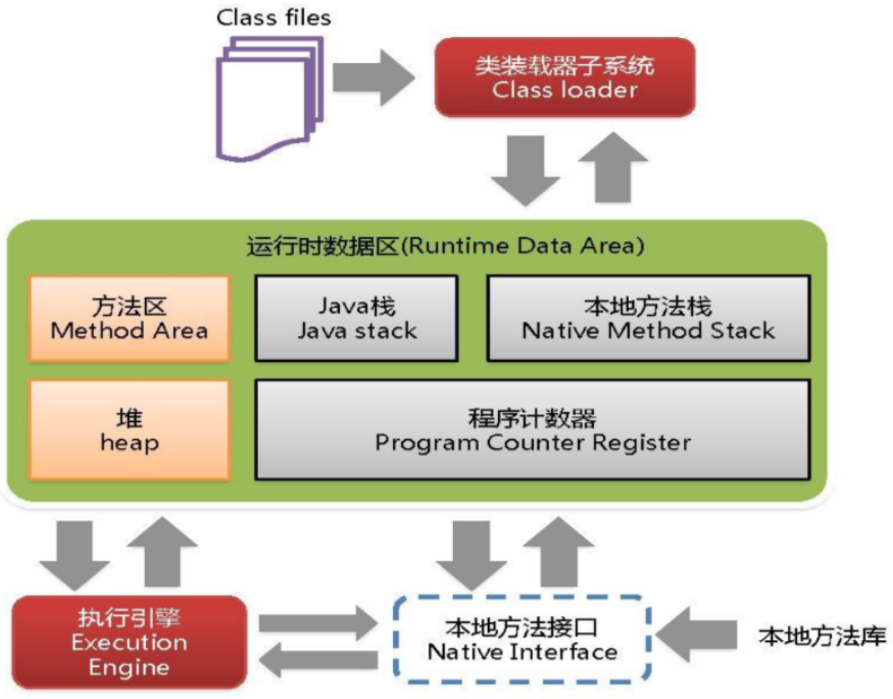
- SUN Classic
- Exact VM
- HotSpot VM： HotSpot指热点代码探测技术

- BEA JRockit：(BEA 已被Oracle收购) 专注于服务端应用，世界最快的jvm之一
- IBM J9
- Taobao JVM: 目前已经在淘宝、天猫上线，替换了Oracle官方JVM；
- Graal VM: Oracle 2018年4月公开，口号 Run Programs Faster Anywhere.最可能替代HotSpot的产品

Android虚拟机 DVM

- 谷歌开发，基于Android，在2.2中提供了JIT
- 只能称作虚拟机 不能称为java虚拟机，他没有遵循Java虚拟机规范
- 基于寄存器架构，效率高，但是跟硬件耦合度比较高
- 不能直接执行class文件，执行的是dex文件
- 5.0使用支持提前编译的ART VM替换Dalvik VM

JVM体系结构概览



1.方法区和堆区是所有线程共享的内存区域；而java栈、本地方法栈和程序员计数器是运行是线程私有的内存区域。

2.Java栈又叫做jvm虚拟机栈

3.方法区（永久代）在jdk8中又叫做元空间Metaspace

- 方法区用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译器（JIT编译器，英文写作Just-In-Time Compiler）编译后的代码等数据。虽然Java虚拟机规范把方法区描述为堆的一个逻辑部分，但是它却有一个别名叫做 Non-Heap（非堆），目的应该是与 Java 堆区分开来。
- 在JDK1.7之前运行时常量池逻辑包含字符串常量池存放在方法区, 此时hotspot虚拟机对方法区的实现为永久代
- 在JDK1.7 字符串常量池被从方法区拿到了堆中, 这里没有提到运行时常量池,也就是说字符串常量池被单独拿到堆,运行时常量池剩下的东西还在方法区, 也就是hotspot中的永久代
- 在JDK1.8之后JVM 已经将运行时常量池从方法区中移了出来，在 Java 堆（Heap）中开辟了一块区域存放运行时常量池。同时在 jdk 1.8中移除整个永久代，取而代之的是一个叫元空间（Metaspace）的区域

4.java代码执行流程：

java程序--（编译javac）-->字节码文件.class-->类装载子系统化身为反射类Class--->运行时数据区--->（解释执行）-->操作系统（Win，Linux，Mac JVM）

所有线程, 共享 堆和 方法区

独享自己的 java栈 和 本地方法栈 和 程序计数器;

栈的指令集架构和寄存器的指令集架构

由于跨平台的设计，java的指令都是根据栈来设计的，不同平台CPU架构不同，所以不能设计为基于寄存器的

二者区别：

栈：跨平台性、指令集小、指令多；执行性比寄存器差
寄存器：指令少

一些简单查看命令：

```
//查看指令集命令代码
cd out/production/类根目录

//反编译
javap -v StackStruTest.class

//打印程序执行的进程
jps

```

jvm生命周期

1.启动

通过引导类加载器（bootstrap class loader）创建一个初始类（initial class）来完成的，这个类是由虚拟机的具体实现指定的.

2.执行

- 一个运行中的java虚拟机有着一个清晰的任务：执行Java程序；
- 程序开始执行的时候他才运行，程序结束时他就停止；
- 执行一个所谓的Java程序的时候，真真正正在执行的是一个叫做Java虚拟机的进程。

3.退出

- 程序正常执行结束
- 程序异常或错误而异常终止
- 操作系统错误导致终止
- 某线程调用Runtime类或System类的exit方法，或Runtime类的halt方法，并且java安全管理器也允许这次exit或halt操作
- 除此之外，JNI规范描述了用JNI Invocation API来加载或卸载Java虚拟机时，Java虚拟机的退出情况