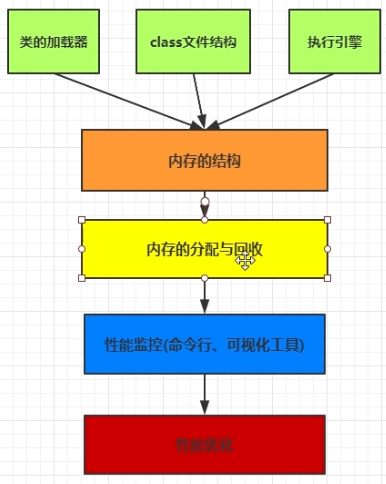
# 学习路线



类加载子系统-----运行时数据区-----执行引擎-----垃圾回收

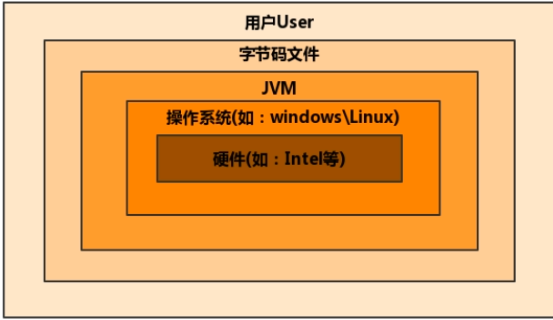
学习路线图

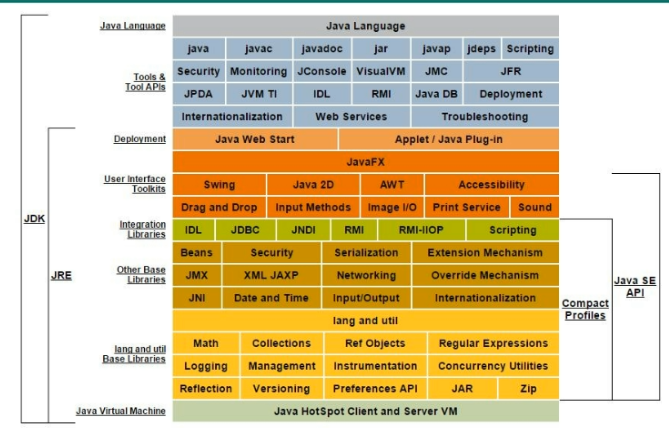


# jvm的特点

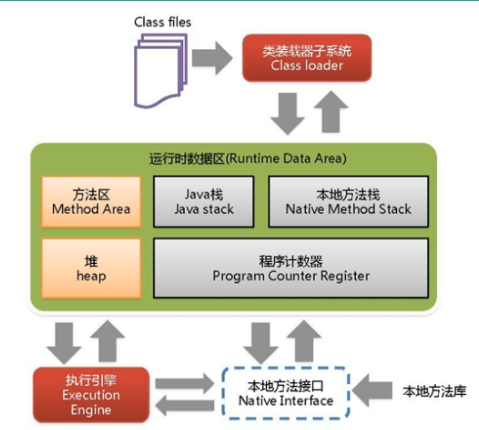
1. 一次编译，到处运行
2. 自动内存管理
3. 自动垃圾回收功能

# 3、概述



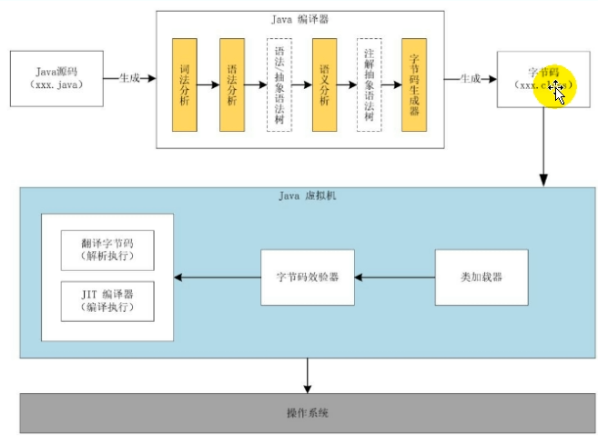


# 4、JVM整体结构



字节码文件 --- 类装载子系统---运行时数据区----执行引擎/本地方法接口

# 5、代码执行流程



# Jvm架构模型

Java编译器输入的指令流基本上是一种基于**栈的指令集架构，**另外一种是基于寄存器的指令集架构。

区别：

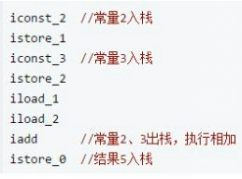
·基于栈的指令集架构的特点

~设计和实现更简单，适用于资源受限的系统（比如机顶盒等嵌入式平台之类）

~避开了寄存器的分配难题，使用零地址指令方式分分配。

~指令流中的指令大部分是**零地址**指令，起执行过程依赖于操作数栈，**指 令集更小，但是完成一项操作所需的指令多**

~不需要硬件支持，可移植性更好，更好实现跨平台

 （java 字节码指令 基于栈）

·基于寄存器的指令集架构特点

~典型的应用是 x86的二进制指令集，比如传统的pc。

~**指令集架构则完全依赖硬件，可以移植性差**

~**性能优秀和执行高效**

~可以使用更少的指令去完成一项操作

~大部分是一地址，二地址，三地址指令

# 7、JVM的生命周期

## 1、虚拟机的启动：

Java 虚拟机的启动时通过引导类加载器（bootstrap class loader）创建一个初 始类（initial class）来完成的，这个类是又虚拟机的具体实现指定的。

## 2、虚拟机的执行：

· 一个运行中的java虚拟机有者一个清晰的任务，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、虚拟机的退出：

有如下几种情况：

·程序正常执行结束

·程序在执行过程中遇到了异常或错误而终止

·由于操作系统出现错误而导致java虚拟机进程终止

· 某线程调用Runtime类或System类的exit()方法，或Runtime类的halt(),并且java安全管理器也允许这exit或halt操作。

· JNI （java native interface）规范描述了用JNI Invocation API 来加载或卸载Java虚拟机时，Java虚拟机的推出情况。

# 8、有哪些常用的虚拟机

J9是IBM开发的一个高度模块化的JVM。

JRockit 是BEA开发