# 一、为什么学虚拟机

学习 Java 虚拟机能深入地理解 Java 这门语言

学习虚拟机是为线上排查问题打下基础

# Java语言的前世今生

JDK 与 JRE

JDK 与 J2SE

J2SE 与 Java SE

Java 语言的历史

# Java虚拟机的历史

# 到底什么是Java虚拟机

# 从源代码到机器码

在 JVM 中有三个非常重要的编译器，它们分别是：前端编译器、JIT 编译器、AOT 编译器。

前端编译器，最常见的就是我们的 javac 编译器，其将 Java 源代码编译为 Java 字节码文件。JIT 即时编译器，最常见的是 HotSpot 虚拟机中的 Client Compiler 和 Server Compiler，其将 Java 字节码编译为本地机器代码。而 AOT 编译器则能将源代码直接编译为本地机器码。这三种编译器的编译速度和编译质量如下：

编译速度上，解释执行 > AOT 编译器 > JIT 编译器。

编译质量上，JIT 编译器 > AOT 编译器 > 解释执行。

而在 JVM 中，通过这几种不同方式的配合，使得 JVM 的编译质量和运行速度达到最优的状态。

# 六、字节码与文件结构

# IMG_256

# Java虚拟机内存结构

Java 虚拟机的内存结构是学习虚拟机所必须掌握的地方，其中以 Java 堆的内存模型最为重要，因为线上问题很多时候都是 Java 堆出现问题。因此掌握 Java 堆的划分以及常用参数的调整最为关键。

除了上述所说的六大部分之外，其实在 Java 中还有直接内存、栈帧等数据结构。但因为直接内存、栈帧的使用场景还比较少，所以这里并不做介绍，以免让初学者一时间混淆。

学到这里，一个 Java 文件就加载到内存中了，并且 Java 类信息就会存储在我们的方法区中。如果创建对象，那么对象数据就会存放在 Java 堆中。如果调用方法，就会用到 PC 寄存器、Java 虚拟机栈、本地方法栈等结构。

# JVM类加载机制

分析一个类的执行顺序大概可以按照如下步骤：

\*\*确定类变量的初始值。\*\*在类加载的准备阶段，JVM 会为类变量初始化零值，这时候类变量会有一个初始的零值。如果是被 final 修饰的类变量，则直接会被初始成用户想要的值。

\*\*初始化入口方法。\*\*当进入类加载的初始化阶段后，JVM 会寻找整个 main 方法入口，从而初始化 main 方法所在的整个类。当需要对一个类进行初始化时，会首先初始化类构造器（），之后初始化对象构造器（）。

\*\*初始化类构造器。\*\*JVM 会按顺序收集 类变量的赋值语句、静态代码块，最终组成类构造器由 JVM 执行。

\*\*初始化对象构造器。\*\*JVM 会按照收集成员变量的赋值语句、普通代码块，最后收集构造方法，将它们组成对象构造器，最终由 JVM 执行。

如果在初始化 main 方法所在类的时候遇到了其他类的初始化，那么就先加载对应的类，加载完成之后返回。如此反复循环，最终返回 main 方法所在类。

# 垃圾回收机制

现今的 Java 虚拟机判断垃圾对象使用的是：GC Root Tracing 算法。其大概的过程是这样：从 GC Root 出发，所有可达的对象都是存活的对象，而所有不可达的对象都是垃圾。

可以看到这里最重要的就是 GC Root 这个集合了，其实 GC Root 就是一组活跃引用的集合。但是这个集合又与一般的对象集合不太一样，这些集合是经过特意筛选出来的，通常包括：

* 所有当前被加载的 Java 类
* Java 类的引用类型静态变量
* Java类的运行时常量池里的引用类型常量
* VM的一些静态数据结构里指向GC堆里的对象的引用
* 等等

简单地说，GC Root 就是经过精心挑选的一组活跃引用，这些引用是肯定存活的。那么通过这些引用延伸到的对象，自然也是存活的。

## **如何进行垃圾回收？**

垃圾回收算法简单地说有三种算法：标记清除算法、复制算法、标记压缩算法。

