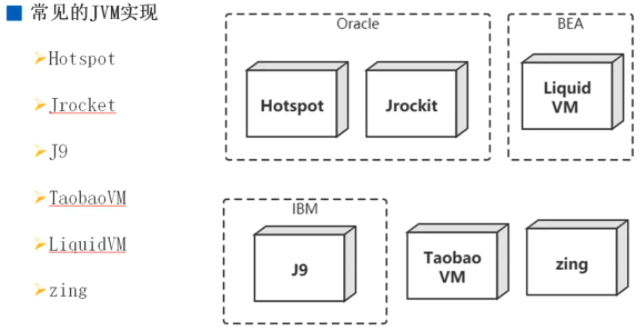
# JVM

## 常见的JVM实现：



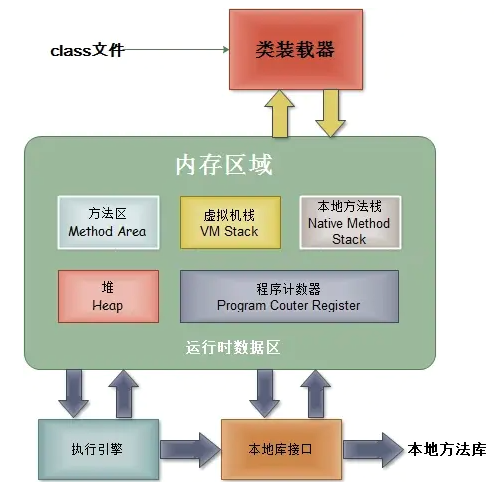
## JVM结构：

类加载器

内存区域

执行引擎

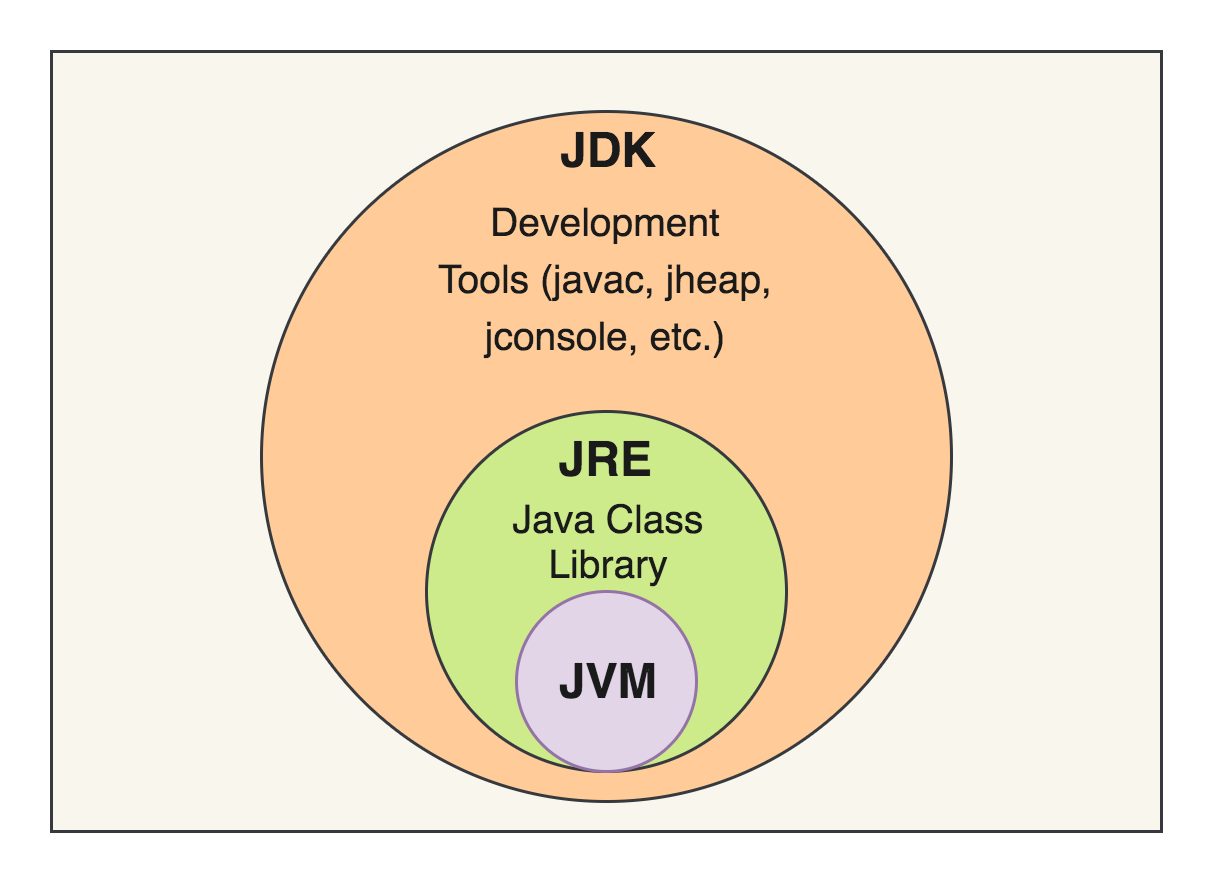
本地库接口



## JVM/JRE/JDK关系：

JDK=JRE+相关开发工具

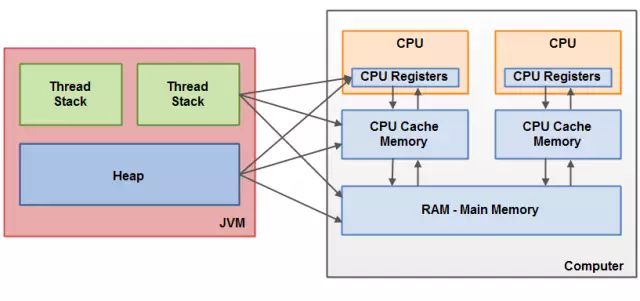
JRE=JVM+核心类库



# JMM

## 背景：

结合CPU的多级缓存架构，Java的线程运行在不同的core上，每个线程有自己的工作内存。



## 带来的问题：

可见性

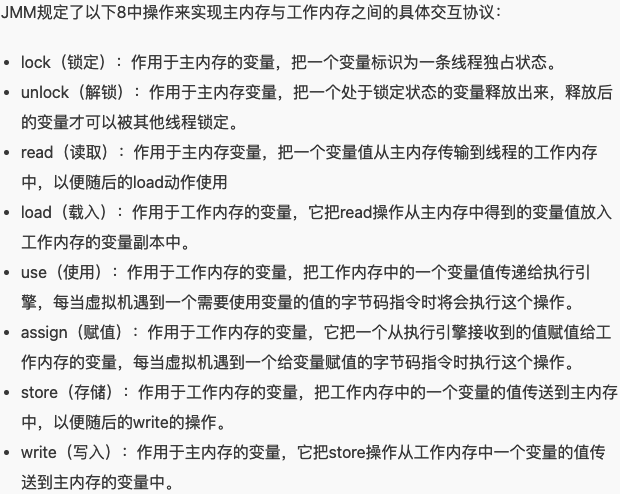
解决方式：volatile，底层lock，可以实现刷主存

## 注意：

原子性，指令重排序 不是多级缓存带来的问题

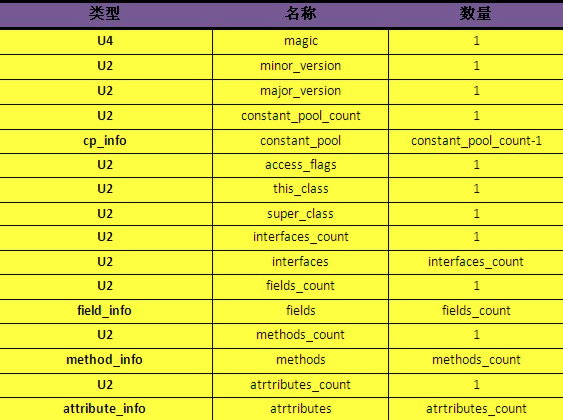
## 原语：

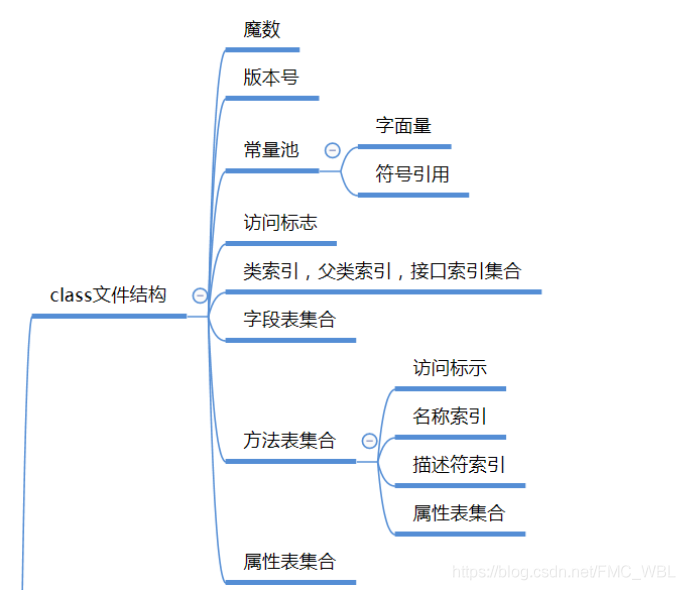
问题：lock和volatile的实现中的lock不是同一个吧？



# Class文件结构

<https://www.jianshu.com/p/ae3f860499aa>





## 1:魔鬼数字+版本

是固定值：0xCAFEBABE，如果不是这个，那么加载会失败

## 2:常量池

存放常量信息（static，string），和符号引用的

## 3:方法信息

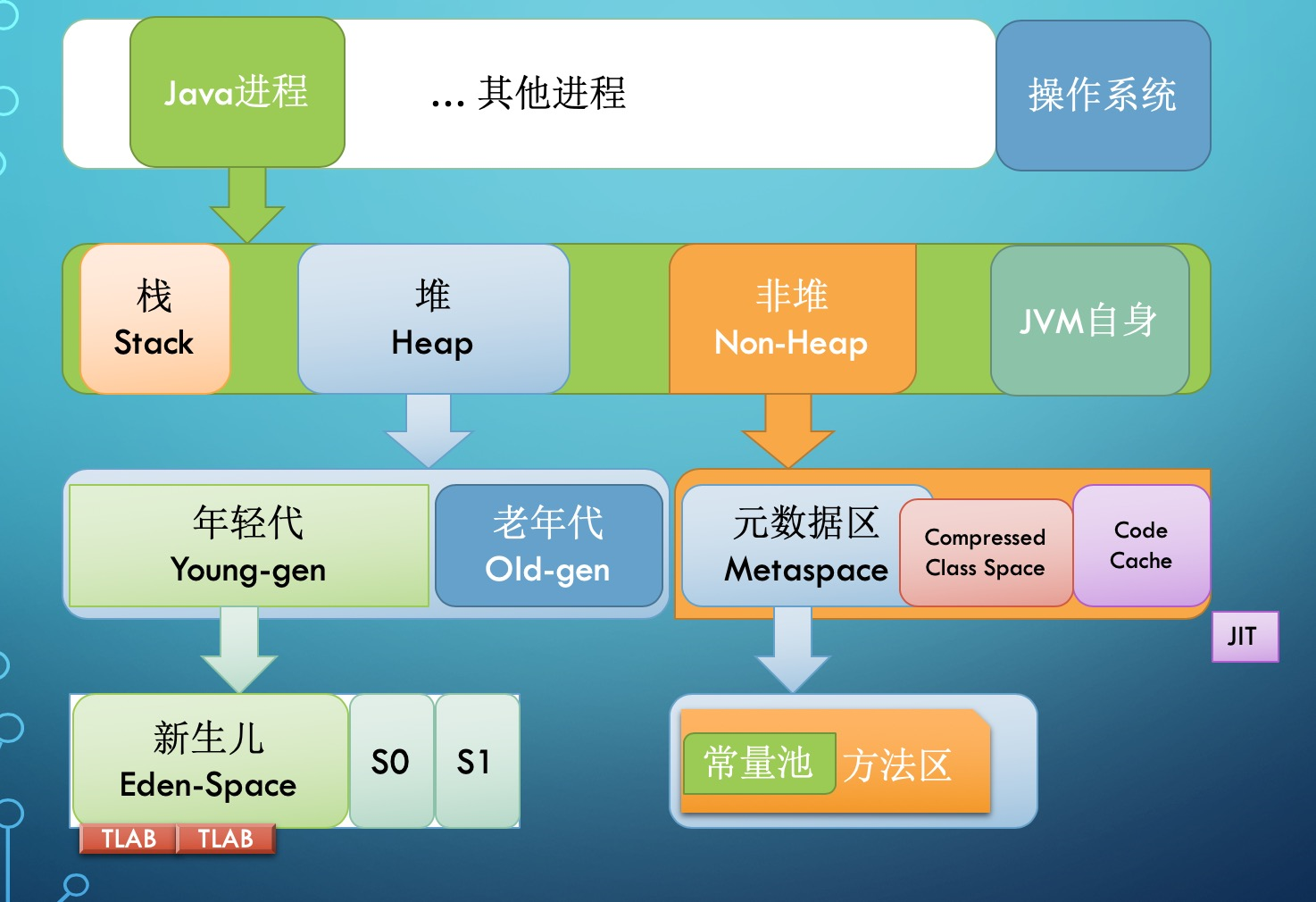
主要是方法的元数据，方法的内容

## 4:字段信息

## 5:类自身的信息

# 运行时数据数据区

## 总览：



## 主要分为：

### 1:堆（线程共享）

存储对象的地方，1.8之后还存储字符串常量

### 2:方法区（线程共享）

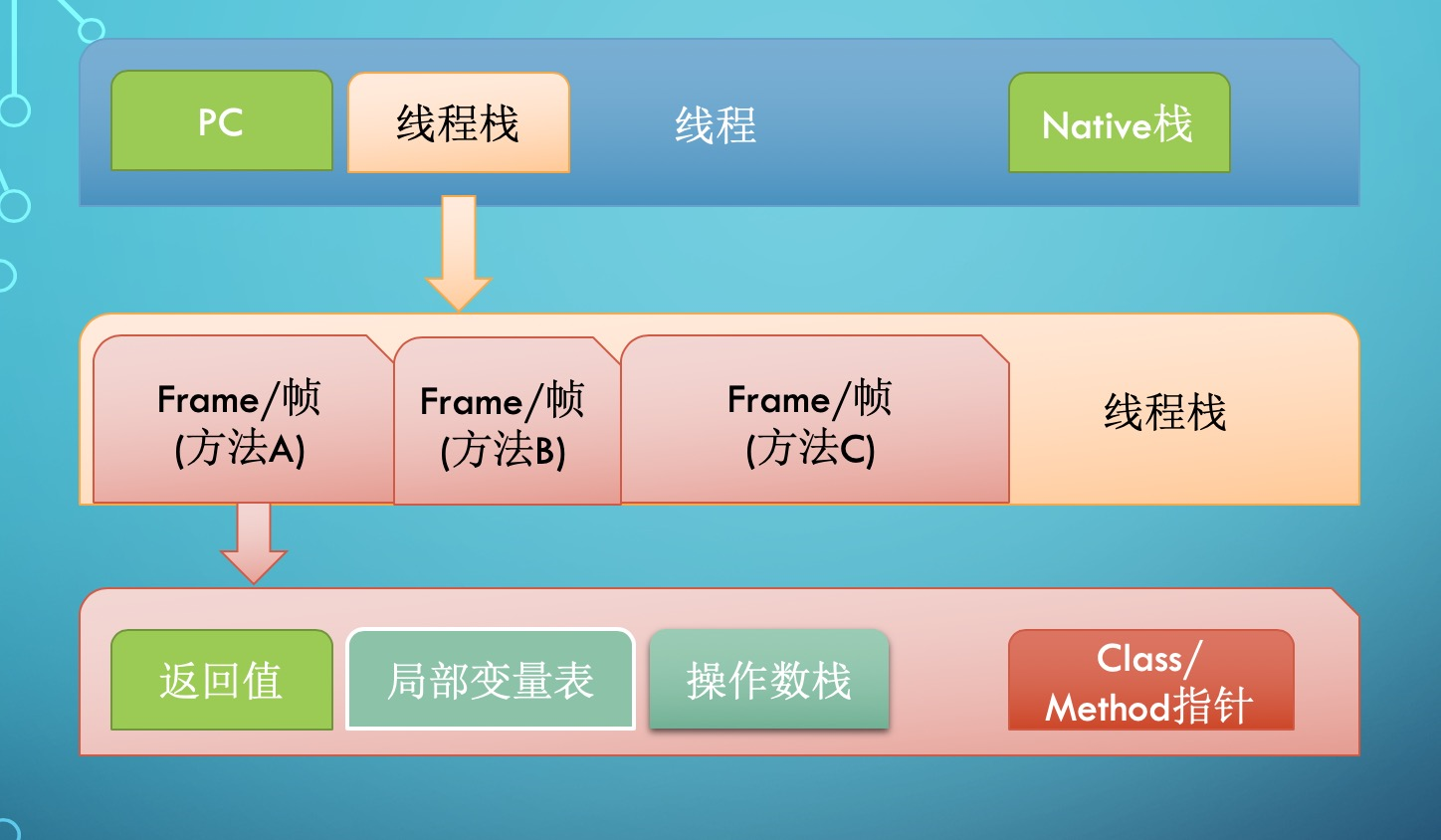
存储class二进制信息的地方（常量池就在这里）

类加载器加载的class文件的二进制就存储于此

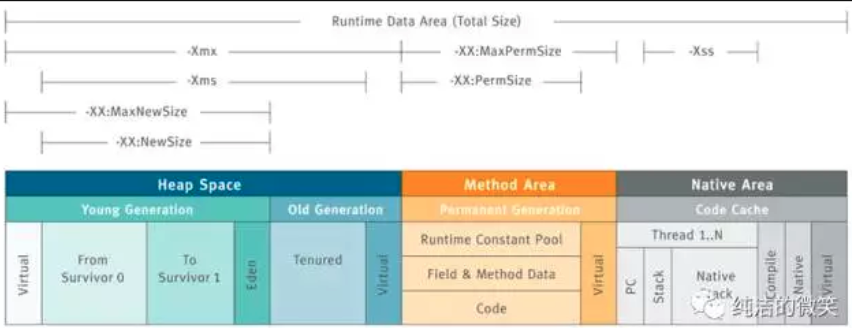
###### 注意：1.7以及之前，方法区叫永久代，容量有限。1.8之后叫元数据区，容量无上限。还有1.7以及之前，string常量存储在方法区，1.8以及之后存储在堆中。

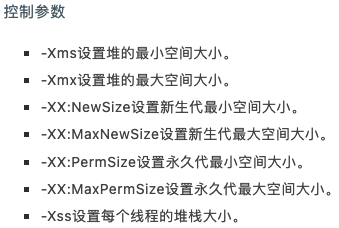
### 3:栈+PC（线程私有）

注意：一个方法对应一个栈帧



## 相关参数：





# 类加载过程

## 前置知识：

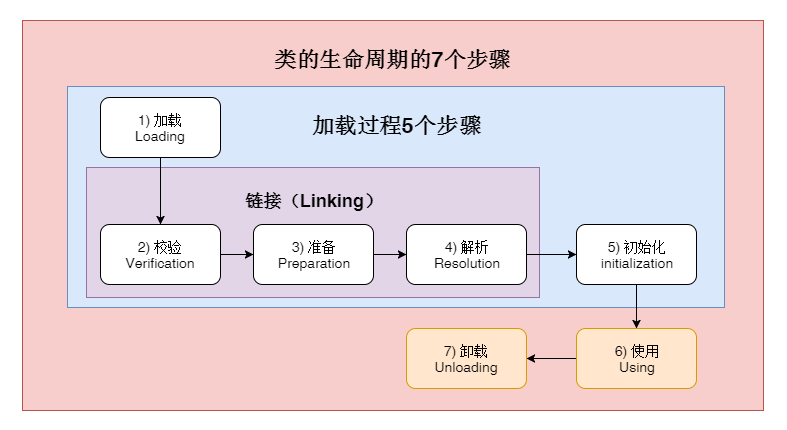
由前面知识可以知道，class文件由**类加载器**加载到运行时区域中的 方法区中。Load之后，在内存中创建2份内容：

1:把二进制内容加载到方法区中（1.7 永久代）（1.8元数据区）

2:生成一个Class对象于堆中，该对象指向方法区中的类内容，方便使用

加载到哪里（2块）

## 类加载过程：



类加载一共分为3个步骤：

### 1:加载：

把xxx.class**文件加载到元数据区**，并**生成class对象于堆中**

### 2:链接

验证：验证class文件格式

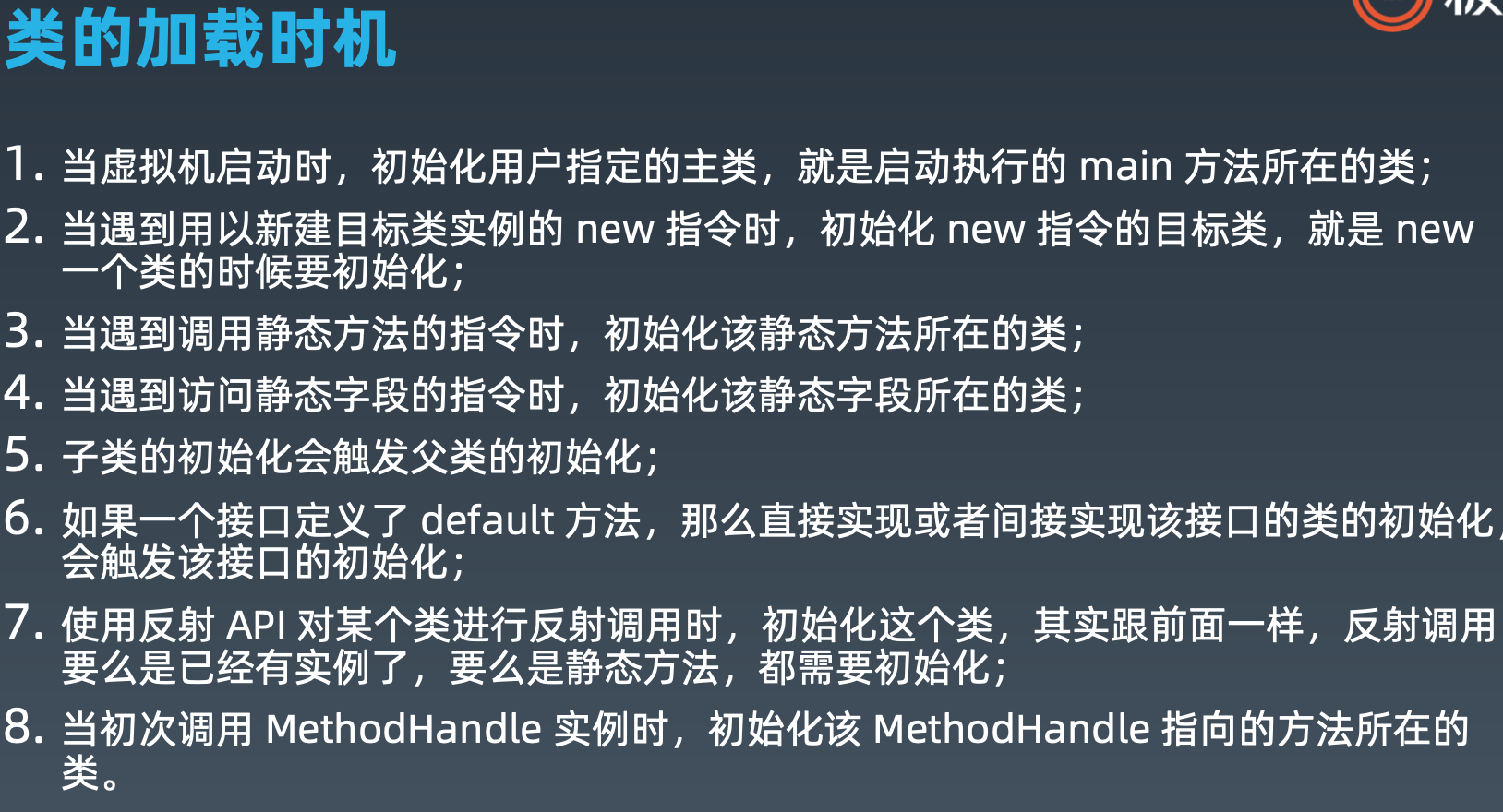
准备：给static变量分配内存，并初始化默认值

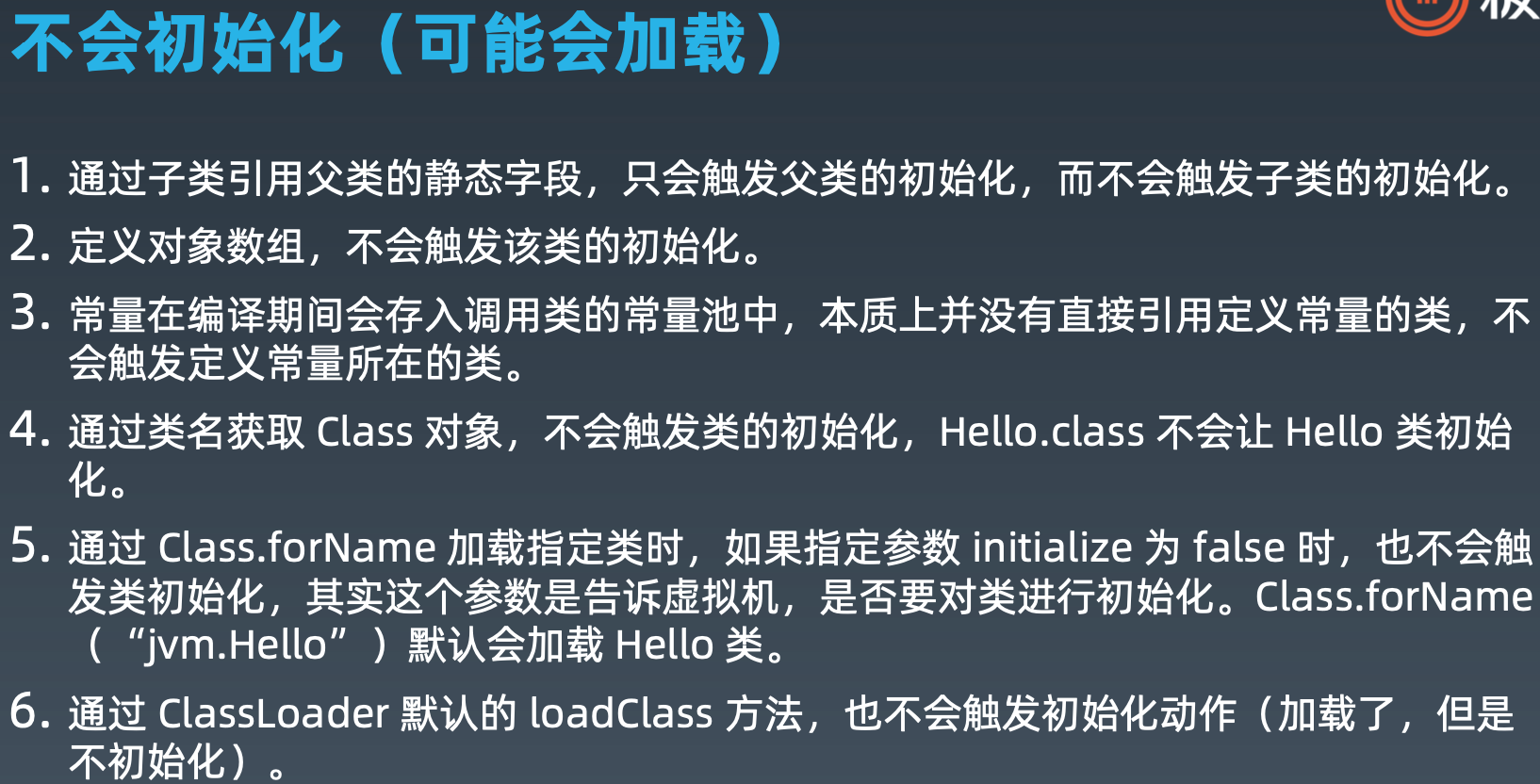
解析：符号引用 --> 直接引用

### 3:初始化

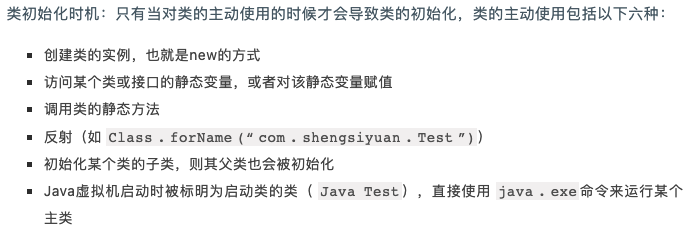
静态变量，静态代码块初始化

## 类加载时机（不要死记硬背）：

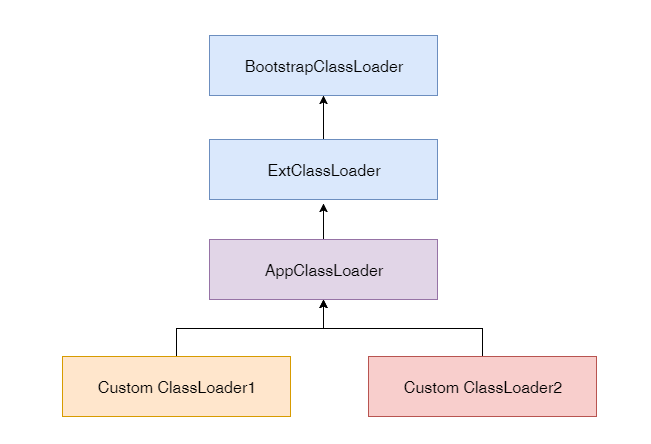




## 类初始化时机（不要死记硬背）：



## 类加载器类型：



注意：类加载器之间的关系不是继承，而是组合

### 1:系统类加载器

C++实现，加载系统基础类，比如String.class，rt.jar，或者ExtClassLoader和AppClassLoder

### 2:扩展类加载器

用来加载jre/lib/ext下面的jar

### 3:App类加载器

应用使用的类加载器，用来加载 classpath中的类

### 4:自定义类加载器

我们自己定义的类加载器，来加载一些特殊的类，比如从网络中加载类，或者是加密的类。

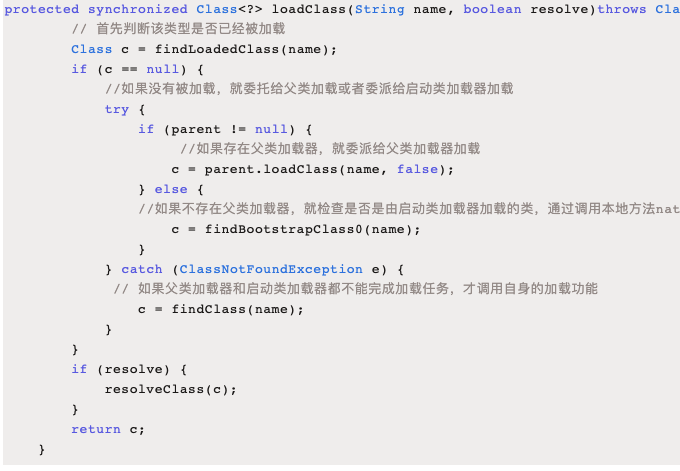
## 双亲委派：

1:先在本地缓存找类是否存在

2:不存在委托父亲找，父亲也是找本地缓存，找不到也是委托父亲

3:父亲没有父亲了，并且缓存也米有，那么委托让孩子加载

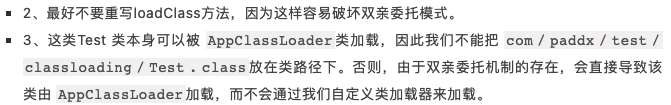
4:孩子看是不是应该自己加载，如果不是，那么让孩子自己加载



## 自定义类加载器方式：

重写findClass





# JMM

JMM的

java -XX:+PrintCommandLineFlags -Xmn10M -Xms40M -Xmx60M -XX:+PrintGC HelloGC

<https://blog.csdn.net/qq_41723615/article/details/104373362>

<https://www.zhihu.com/question/41922036>