类加载

类生命周期

- 加载
  - 找 Class 文件：获取二进制classfile格式的字节流
- 验证
  - 验证格式、依赖：校验范围主要包括文件格式、编译版本号、常量池符号和类型
- 准备
  - 静态字段、方法表：创建静态变量并赋初值、如果静态变量是final修饰的话会直接赋值、分配方法表,即在方法区中分配这些变量所使用的内存空间
- 解析
  - 解析常量池中的符号引用转为直接引用，包括类或接口的解析、字段解析、类方法解析、接 口方法解析

链接

- 初始化
  - 执行构造函数、执行静态代码块、静态变量赋初值
- 使用
- 卸载

类加载的时机

- 虚拟机启动的时候初始化指定的主类，启动执行main方法的类
- new一个类的时候，该类会进行初始化
- 调用类的静态方法，该类会被加载
- 调用类的静态成员变量，该类会被加载
- 子类初始化的时候触发父类也被初始化
- 定义了default方法的接口，直接或间接实现了该接口的类初始化的时候会触发该类初始化
- 使用反射创建实例的时候，会进行初始化（和new一样）
- 当初次调用 MethodHandle 实例时，初始化该 MethodHandle 指向的方法所在的类

类不会初始化可能会加载

- 通过子类引用父类的静态字段，父类不会初始化
- 定义对象数组，该类不会初始化
- 常量在编译期间会存入调用类的常量池中，本质上并没有直接引用定义常量的类，不会触发定义常量所在的类的初始化
- 通过类名获取Class对象，不会触发类的初始化
- 通过 Class.forName 加载指定类时，如果指定参数 initialize 为 false 时，也不会触发类初始化，其实这个参数是告诉虚拟机，是否要对类进行初始化
- 通过 ClassLoader 默认的 loadClass 方法，也不会触发初始化动作（加载了，但是不初始化）

类加载器

- 启动类加载器BootstrapClassLoader
  - 加载java核心类库，如jre/lib目录下的rt.jar。它是由C++编写
- 扩展类加载器ExtClassLoader
  - 加载扩展目录下的类（lib/ext或者java.ext.dirs属性指定的目录）
- 应用类加载器AppClassLoader
  - 加载classpath目录、java.class.path系统属性指定目录下的类
- 自定义类加载器

类加载机制

- 双亲委派
  - 当一个自定义类加载器需要加载一个类，比如java.lang.String，它很 懒，不会一上来就直接试图加载它，而是先委托自己的父加载器去加载，父加载 器如果发现自己还有父加载器，会一直往前找，这样只要上级加载器，比如启动 类加载器已经加载了某个类比如java.lang.String，所有的子加载器都不需要自己 加载了。如果几个类加载器都没有加载到指定名称的类，那么会抛出ClassNotFountException异常
- 负责依赖
  - 如果一个加载器在加载某个类的时候，发现这个类依赖于另外几个类 或接口，也会去尝试加载这些依赖项
- 缓存加载
  - 为了提升加载效率，消除重复加载，一旦某个类被一个类加载器加 载，那么它会缓存这个加载结果，不会重复加载