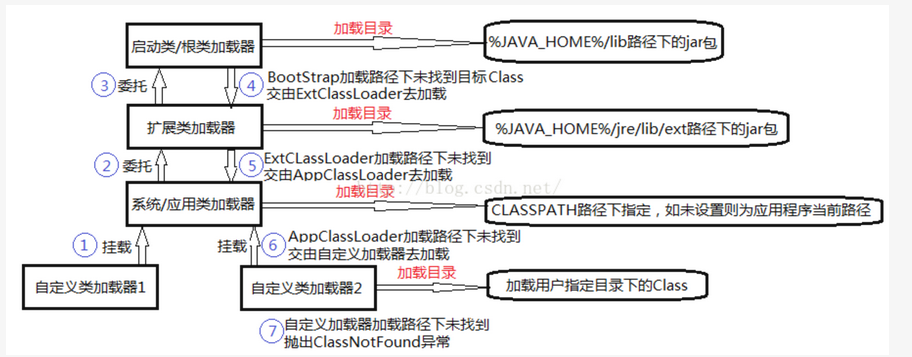
1、编写的java类是如何被加载进内存的？

Java类(.java)文件经过javac编译成字节码文件.class。

Jre的加载器从硬盘中读取.class字节码文件，载入系统，分配给JVM的内存区域—运行数据区（Runtime Data Areas）,然后再执行引擎解释或者编译 类文件， 转化成特定的CPU的机器码，CPU执行机器码，完成整个加载过程，即程序启动？

2、类加载器是如何进行层级划分的？

类加载器被组织成一种层级结构关系，也就是父子关系。其中，Bootstrap是所有类加载器的父亲。



BootStrap ClassLoader(启动类/根类加载器)：运行java虚拟机时，该加载器类被创建，会加载一些基本的java API,包括Object类。需注意的是，该类不是用java语言编写的，而是由C/C++编写的。【负责加载JDK中的核心类库】【当JVM启动后，Bootstrap ClassLoader也随着启动，负责加载完核心类库后，并构造Extension ClassLoader和App ClassLoader类加载器。】

Extension ClassLoader(扩展类加载器): 加载出了API之外的扩展类，包括一些与安全性能相关的类

AppClassLoader(系统/应用类加载器): 它加载应用程序中的类，也就是在你的classpath中配置的类。(加载的是classpath下的class文件)。

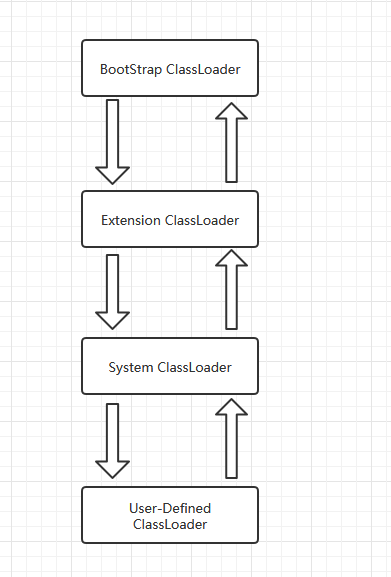
User-Defined ClassLoader(自定义加载加载器): 这是开发人员通过拓展ClassLoader类定义的自定义加载器，加载自定义的一些类。【都必须继承自java.lang.ClassLoader类】

3、类加载器的双亲委派模式（Delegation Mode）？

当JVM加载一个类的时候，下层的加载器会将任务委托给上层类加载器，上一层加载器会检查它的命名空间中是否已经加载了这个类，如果已经加载了，直接使用这个类。如果没有加载，继续往上委托。检查完了之后，按照相反的顺序进行加载，如果Bootstrap加载器找不到这个类，则往下委托，知道找到这个类文件。对于特定的类加载器来说，一个java类只能被载入一次，也就是说java虚拟机中，类的完整标识是（ClassLoader、package、className）。

java.\*开头的类，jvm的实现中已经保证了必须由bootstrp来加载。（classLoader，package，

而程序在启动的时候，并不会一次性加载程序所要用的所有class文件，而是根据程序的需要，通过Java的类加载机制(ClassLoader)来动态加载某个class文件到内存当中的，从而只有class文件被载入到了内存之后，才能被其它class所引用。所以ClassLoader就是用来动态加载class文件到内存当中用的。

className）。

4、为什么要使用双亲委派模式？

一是可以避免重复加载，二是安全，避免自定义的类替代java核心api中定义的类。

自下而上的委托，自上而下的加载。

5、但是JVM在搜索类的时候，又是如何判定两个class是相同的呢？

答：JVM在判定两个class是否相同时，不仅要判断两个类名是否相同，而且还要判断是否由同一个类加载器实例加载的。只有两个同时满足的情况下，JVM才认为这两个class是相同的。就算两个class是同一份class字节码，如果被两个不同的ClassLoader实例加载，JVM也会认为它们是两个不同的class(类)。

6、如上所说，类加载器是自下而上的委派，自上而下的加载，那么是如何做到一个类被多个类加载器加载？当类被第一个类加载器加载以后，不是无法被其他类加载器所加载吗？

答：自上而下的加载，是JVM中存在的类加载器层级，一共分为四层加载器：

根类加载器 》扩展类加载器》应用类加载器》自定义加载器

前三个为jvm中存在的加载器，随着启动而创建。只有一个

但是相应的自定义加载器，只要继承了java.lang.ClassLoader类就行，所以可以自定义无数个该类型的类加载器，自定义的类加载器没有层级的划分，所以可以多个类加载器加载同一个class.

比如网络上的一个Java类org.classloader.simple.NetClassLoaderSimple，javac编译之后生成字节码文件NetClassLoaderSimple.class，ClassLoaderA和ClassLoaderB这两个类加载器并读取了NetClassLoaderSimple.class文件，并分别定义出了java.lang.Class实例来表示这个类，对于JVM来说，它们是两个不同的实例对象，但它们确实是同一份字节码文件，如果试图将这个Class实例生成具体的对象进行转换时，就会抛运行时异常java.lang.ClassCaseException，提示这是两个不同的类型。现在通过实例来验证上述所描述的是否正确：