<https://www.cnblogs.com/wangenxian/p/10936640.html>

Class类包含了类的信息，如构造方法、方法、属性，可用于反射。



获取Class类对象的几种方法

Test test = new Test();

(1).test.getClass();

在运行时确定，所以运行实例才是该类对象。super.getClass()不能获得父类的类对象，仍然是当前类对象。

获得父类类对象： test.getClass().getSuperclass()

(2).Test.class;

在编译时确定，返回当前类的类对象实例。不会加载静态变量

获得父类类对象： Test.class.getSuperclass()

(3).Class.forName("类路径");会加载静态变量，比如jdbc驱动利用这个加载一些静态属性

通过静态方法获取类对象，Class.forName("com.wan.Test");

在运行期间，如果我们要产生某个类的对象，Java虚拟机(JVM)会检查该类型的Class对象是否已被加载。如果没有被加载，JVM会根据类的名称找到.class文件并加载它。一旦某个类型的Class对象已被加载到内存，就可以用它来产生该类型的所有对象

普通类为什么不能用static修饰？

讲一下个人理解吧。static修饰的东西被我们成为类成员,它会随着类的加载而加载,比如静态代码块,静态成员,静态方法(这里只是加载,并没有调用)等等,可以假象一下,如果把一个Class文件中的外部类设为static,那目的何在呢？难道让这个类随着应用的启动而加载吗？如果我在这次使用过程中根本没有使用过这个类,那么是不是就会浪费内存。这样来说设计不合理,总而言之,设计不合理的地方,Java是不会让它存在的。

而为什么内部类可以使用static修饰呢,因为内部类算是类的成员了,如果我们没有使用静态来修饰,那么我们在创建内部类的时候就需要先有一个外部类的对象,如果我们一直在使用内部类,那么内存中就会一直存在外部类的引用,而我们有时候只需要使用内部类,不需要外部类,那么还是会浪费内存,甚至会造成内存溢出。使用static修饰内部类之后,内部类在创建对象时就不需要有外部类对象的引用了。

最终结论就是:static可以用来修饰内部类,但是不可以用来修饰外部类

classLoader相关理解

我们都知道java程序写好以后是以.java（文本文件）的文件存在磁盘上，然后，我们通过(bin/javac.exe)编译命令把.java文件编译成.class文件（字节码文件），并存在磁盘上。但是程序要运行，首先一定要把.class文件加载到JVM内存中才能使用的，我们所讲的classLoader,就是负责把磁盘上的.class文件加载到JVM内存中，如下图所示：



你可以认为每一个Class对象拥有磁盘上的那个.class字节码内容,每一个class对象都有一个getClassLoader()方法，得到是谁把我从.class文件加载到内存中变成Class对象的。