**1、类加载器classLoad**

将.java文件编译成.class文件加载到Java虚拟机中。

Java自带的类加载器

1. BootStrapClassLoader 隐式存在，他不存在jdk里面，他存在jvm里面，这个对象是看不到的。
2. ExtClassLoader 这个对象看的到，扩展类加载器，主要加载ext包下面的文件
3. ApplicationClassLoader 系统上下文class文件，项目里面的所有Class

知识点：

（1）当类加载器去加载Class文件的时候，默认情况下并不会由自己去加载，而是由父类加载器去加载（双亲委派，如果父类返回空，才由他的子类处理，也就是BootStrapClassLoader永远是第一个进行处理的），上面就是类加载器的父子关系。

优点：

保证系统里面同一个Class对象只会存在一份

保证系统核心的class文件不被篡改

（2）他们是这样做继承关系的：

Class ApplicationClassLoader{

ExtClassLoader parent;

}

（3）同一个Class被不同的ClassLoader加载，他们的类型是不相同的。比如：

A a = classloaderA.load

A b = classloaderB.load

a = b（报错，类型不相同）

（4）ClassLoader.loadClass 是从缓存中获取Class对象

（5）ClassLoader.defineClass 把Class文件转换成Class对象，并放到缓存中

（6）全盘委托：用去确认默认使用的类加载器（new关键字用那个类加载器），当有new关键字需要类加载器加载的时候，jvm会判断当前调用new关键字的类加载器是什么，然后用调用new关键字的类加载器加载，简单的就是说，new Test()使用的类加载器是main方法的类加载器

**2、JIT及时编译**

java的编译机制的时候，分为两个部分。

一个是javac指令将java源码变为java字节码的静态编译过程。另一个是java字节码编译为本地机器码的过程，并且因为这个过程是在程序运行时期完成的所以称之为即时编译。

**3、JVM（java虚拟机）内存区域（JVM不是java写的，是openJDK写的）**

方法区（信息共享，其他区可以随时使用）：

存放类信息（.class文件）、常量、static变量、JIT编译后的代码

java堆区（信息共享，其他区可以随时使用）：

存放实例对象（同时也会本地缓存一个Class引用，当类加载器去读取的对象的时候，会先去缓存里找引用），GC（垃圾回收器）主要操作这个区域

VM stack（虚拟机栈）：

Java方法在运行时的内存模型，用来存放栈贞的一个区域。栈贞里面存放的是局部变量、引用类型数据的地址、操作数栈

PC（程序计数器）：

Java线程的私有数据，这个数据就是执行下一条指令的地址。PC的主要功能是用于存放指令的地址

Native method stack（本地方法栈）：Java线程的私有数据，与JVM虚拟机的native方法有关。

4、Java内存模型 Java memory model JMM（规范，抽象的模型）：

(1)主内存：共享信息

(2)工作内存：私有信息

(3)工作方式：

现场修改私有数据，直接在工作空间修改

现场修改共享数据，把数据复制到工作空间中，在工作空间中修改，修改完成以后，刷新内存中的数据

Java内存和Java虚拟机内存的关系：Java内存是物理内存，Java虚拟机内存是逻辑内存。物理上的内存划分出Java虚拟机逻辑上的内存，然后在不同的逻辑内存里面完成不同的事情