https://www.cnblogs.com/chanshuyi/p/jvm\_serial\_07\_jvm\_class\_loader\_mechanism.html

**类的加载机制：**

加载、验证、准备、解析、初始化、使用、卸载。

加载：加载代码到内存中（加载class字节码文件）。

验证：验证文件是否符合jvm规范，验证代码是否有语法等错误。

准备：为static修饰的对象/方法分配内存，但不初始化。static final修饰的分配内存并初始化。

解析：JVM 针对类或接口、字段、类方法、接口方法、方法类型、方法句柄和调用点限定符 7 类引用进行解析，将其在常量池中的符号引用替换成直接其在内存中的直接引用。

初始化：初始化类：按顺序初始化static修饰的。

初始化(new)对象：若类没有进行初始化，则先初始化类，再初始化对象；不初始化static修饰的。

初始化类时先初始化父类。

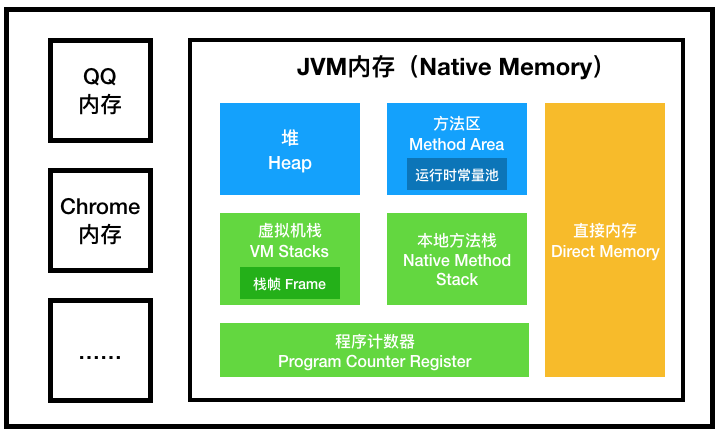
获取父类的静态变量时，只初始化父类。

使用：执行代码。

卸载：销毁class文件。

Spring项目启动时，jvm只加载并初始化在容器内的类及对象，不在容器内的类在用到时才会加载。

**JVM内存模型：**

****

**线程共享部分：java堆，方法区，常量池。**

方法区&常量池：

存储 Java 类字节码数据的一块区域，它存储了每一个类的结构信息，如运行时常量池、类的成员变量、类的方法等。

Java堆：

用于java实例对象的内存分配。Jvm GC指回收java堆的内存。

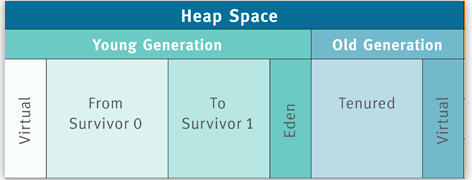
Java堆被分为年轻代和老年代两个区域。需要为对象分配内存时，优先分配在Eden区，Eden区内存不够时会进行GC，年轻代的对象经历过指定次数的GC则会被移动到老年代。年轻代空间大小默认分配Eden:from:to=8:1:1，因为大部分对象的存活时间都比较短，这样分配可以减少内存空间的浪费。

Java堆其实还未每一个线程单独分配了一块TLAB空间，这部分空间在分配时是线程独享的，在使用时是线程共享的。

为了保证内存分配的线程安全，防止出现两个不同对象指向同一个内存区域的情况。

每个线程在Java堆中预先分配一小块内存，然后再给对象分配内存的时候，直接在自己这块”私有”内存中分配，当这部分区域用完之后，再分配新的”私有”内存。（HotSpot虚拟机实现）

另外，TLAB仅作用于新生代的Eden Space，对象被创建的时候首先放到这个区域，但是新生代分配不了内存的大对象会直接进入老年代。因此在编写Java程序时，通常多个小的对象比大的对象分配起来更加高效。



Tips:

虚拟机规范对方法区实现的位置并没有明确要求，在最著名的HotSopt虚拟机实现中（在Java 8 之前），方法区仅是逻辑上的独立区域，在物理上并没有独立于堆而存在，而是位于永久代中（java堆被分为年轻代、老年代、永久代）。所以，这时候方法区也是可以被垃圾回收的。

在Java 8中 ，HotSpot虚拟机移除了永久代，使用本地内存来存储类元数据信息并称之为：元空间（Metaspace）。

**线程私有部分：pc寄存器，java虚拟机栈，native方法栈。**

PC寄存器：保存线程当前正在执行的方法的地址。

Java虚拟机栈：存储当前线程的局部变量表(各种基本数据类型,对象的引用)及操作数栈。

Native方法栈：存储当前线程调用native方法的局部变量表及操作数栈。

**直接内存：**

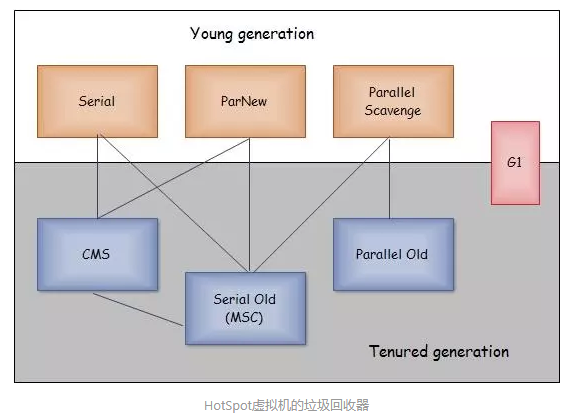
又称堆外内存，这部分内存不由jvm管理和回收，需要手动回收。

直接内存默认空间为-Xmx(最大堆空间)，可以通过-XX:MaxDirectMemorySize设置。

一般使用Unsafe类和NIO中的DirectByteBuffer(实际上还是使用Unsafe)来申请管理回收直接内存。

Jdk1.4中引入了NIO，DirectByteBuffer使用Unsafe类直接申请/回收直接内存。DirectByteBuffer对象指向申请到的直接内存，当处于java堆中的DirectByteBuffer对象被GC回收时，指向的直接内存也将被释放(使用Cleaner实现)。

**JVM垃圾回收器：**

****

新生代GC（Minor GC）：

发生在新生代的垃圾回收动作，Minor GC比较频繁，一般速度也比较快。

老年代GC（Major GC / Full GC）：

发生在老年代的垃圾回收动作，经常会伴随着至少一次的Minor GC。速度较慢。

吞吐量：

CPU用于运行用户代码的时间与CPU总消耗时间的比值，即吞吐量 = 运行用户代码时间 /（运行用户代码时间 + 垃圾收集时间）。垃圾收集时间越短，吞吐量越高。