

# ****程序计数器：（线程私有）****

每个线程拥有一个程序计数器，在线程创建时创建，指向下一条指令的地址

执行本地方法时，其值为undefined。

说的通俗一点，我们知道，Java是支持多线程的，程序先去执行A线程，执行到一半，然后就去执行B线程，然后又跑回来接着执行A线程，那程序是怎么记住A线程已经执行到哪里了呢？这就需要程序计数器了。因此**，为了线程切换后能够恢复到正确的执行位置，每条线程都有一个独立的程序计数器**，这块儿属于“线程私有”的内存。

## ****Java虚拟机栈：（线程私有）****

每个**方法被调用**的时候都会创建一个**栈帧**，用于存储局部变量表、操作栈、动态链接、方法出口等信息。局部变量表存放的是：编译期可知的基本数据类型、对象引用类型。

    每个方法被调用直到执行完成的过程，就对应着一个栈帧在虚拟机中从入栈到出栈的过程。

在Java虚拟机规范中，对这个区域规定了两种异常情况：

　　（1）如果线程请求的栈深度太深，超出了虚拟机所允许的深度，就会出现StackOverFlowError（比如无限递归。因为每一层栈帧都占用一定空间，而 Xss 规定了栈的最大空间，超出这个值就会报错）

　　（2）虚拟机栈可以动态扩展，如果扩展到无法申请足够的内存空间，会出现OOM

## ****本地方法栈：****

（1）本地方法栈与java虚拟机栈作用非常类似，其区别是：java虚拟机栈是为虚拟机执行java方法服务的，而本地方法栈则为虚拟机执使用到的**Native方法服务**。

（2）Java虚拟机没有对本地方法栈的使用和数据结构做强制规定，Sun HotSpot虚拟机就把java虚拟机栈和本地方法栈合二为一。

（3）本地方法栈也会抛出StackOverFlowError和OutOfMemoryError。

## ****Java堆：即堆内存（线程共享）****

（1）堆是java虚拟机所管理的内存区域中最大的一块，java堆是被所有线程共享的内存区域，在java虚拟机启动时创建，堆内存的唯一目的就是存放对象实例几乎所有的对象实例都在堆内存分配。

（2）**堆是GC管理的主要区域**，从垃圾回收的角度看，由于现在的垃圾收集器都是采用的**分代收集**算法，因此java堆还可以初步细分为**新生代和老年代**。

（3）Java虚拟机规定，堆可以处于物理上不连续的内存空间中，只要逻辑上连续的即可。在实现上既可以是固定的，也可以是可动态扩展的。如果在堆内存没有完成实例分配，并且堆大小也无法扩展，就会抛出OutOfMemoryError异常。

**新生代：**

比如我们在方法中去new一个对象，那这方法调用完毕后，对象就会被回收，这就是一个典型的新生代对象。

现在的商业虚拟机都采用这种收集算法来回收新生代，新生代中的对象98%都是“朝生夕死”的，所以并不需要按照1:1的比例来划分内存空间，而是将**内存分为一块比较大的Eden空间和两块较小的Survivor空间**，每次使用Eden和其中一块Survivor。当回收时，将Eden和Survivor中还存活着的对象一次性地复制到另外一块Survivor空间上，最后清理掉Eden和刚才用过的Survivor空间。HotSpot虚拟机默认Eden和Survivor的大小比例是8:1，也就是说，每次新生代中可用内存空间为整个新生代容量的90%（80%+10%），只有10%的空间会被浪费。

当然，98%的对象可回收只是一般场景下的数据，我们没有办法保证每次回收都只有不多于10%的对象存活，**当Survivor空间不够用时，需要依赖于老年代进行分配担保，所以大对象直接进入老年代**。同时，**长期存活的对象将进入老年代（**虚拟机给每个对象定义一个年龄计数器）。

**Minor GC和Full GC：**

GC分为两种：Minor GC和Full GC

**Minor GC：**

　　 Minor GC是发生在新生代中的垃圾收集动作，采用的是复制算法。

对象在Eden和From区出生后，在经过一次Minor GC后，如果对象还存活，并且能够被to区所容纳，那么在使用复制算法时这些存活对象就会被复制到to区域，然后清理掉Eden区和from区，并将这些对象的年龄设置为1，以后对象在Survivor区每熬过一次Minor GC，就将对象的年龄+1，当对象的年龄达到某个值时（默认是15岁，可以通过参数 --XX:MaxTenuringThreshold设置），这些对象就会成为老年代。

但这也是不一定的，对于一些较大的对象（即需要分配一块较大的连续内存空间）则是直接进入老年代

**Full GC：**

　　 Full GC是发生在老年代的垃圾收集动作，采用的是标记-清除/整理算法。

老年代里的对象几乎都是在Survivor区熬过来的，不会那么容易死掉。因此Full GC发生的次数不会有Minor GC那么频繁，并且做一次Full GC要比做一次Minor GC的时间要长。

另外，如果采用的是标记-清除算法的话会产生许多碎片，此后如果需要为较大的对象分配内存空间时，若无法找到足够的连续的内存空间，就会提前触发一次GC。

**老年代：**

    在新生代中经历了N次垃圾回收后仍然存活的对象就会被放到老年代中。而且大对象直接进入老年代。

**永久代：**

    即方法区。

## ****方法区：（线程共享）****

（1）用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据。

（2）Sun HotSpot虚拟机把方法区叫做永久代（Permanent Generation），方法区中最终要的部分是运行时常量池。

（3）运行时常量池是方法区的一部分，自然受到方法区内存的限制，当常量池无法再申请到内存时就会抛出OutOfMemoryError异常。