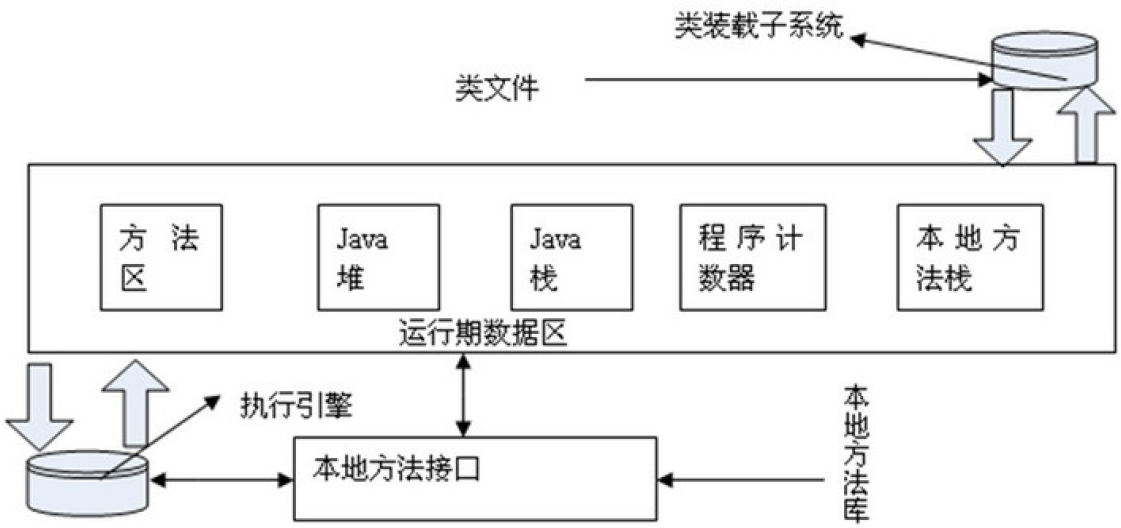
Java运行时数据区



程序计数器

程序计数器（Program Counter Register）是一块较小的内 存空间，它的作用可以看做是当前线程所执行的字节码的行号指示器。(**每个线程都有自己的程序计数器，线程隔离**)

Java虚拟机栈

它描述的是Java 方法执行的内存模型：每个方法被执行的时候都会同时创建一个栈帧（Stack Frame）用于存储局部变量表、操作栈、动态链接、方法出口等信息。**线程私有(线程隔离)**

本地方法栈(线程私有)

本地方法栈（Native Method Stacks）与虚拟机栈所发挥的作用是非常相似的，其区别不过是虚拟机栈为虚拟机执行Java 方法（也就是字节码）服务，而本地方法栈则是为虚拟机使用到的Native 方法服务。

Java堆

此内存区域的唯一目的就是存放对象，一个JVM实例只存在一个堆，堆内存的大小是可以调节的.**堆内存是线程共享的。**totalMemory 默认是系统64分之一 250M　maxMemory 默认是系统的四分之一 4g

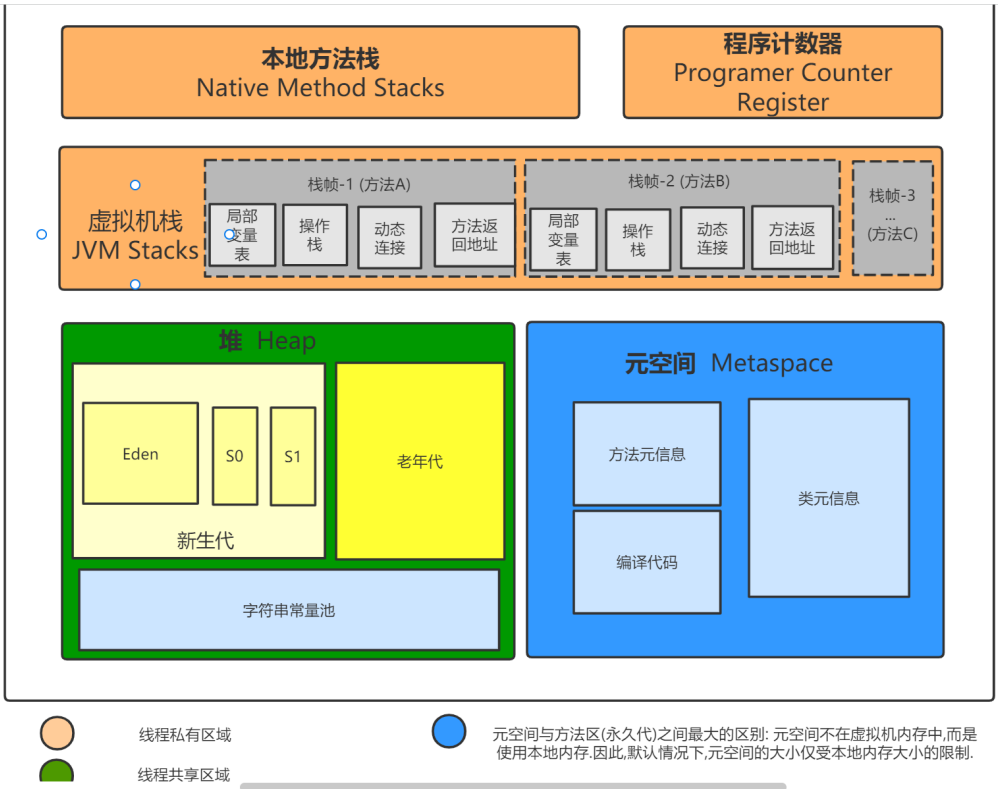
-Xms200m　-Xmx300m

方法区(线程共享)

方法区（Method Area）与Java 堆一样，是各个线程共享的内存区域，它用于存储已被虚拟机加载的类信息、常量 、静态变量等数据。

方法区是逻辑概念

- 永久代
- 元空间



内存管理

显式的内存管理(C/C++)
内存管理(内存的申请和释放)是程序开发者的职责 malloc() free()
常见问题:
内存泄漏:内存空间已经申请,使用完毕后未主动释放
野指针:使用了一个指针,但是该指针指向的内存空间 已经被free
隐式的内存管理(Java/C#)
内存的管理是由垃圾回收器自动管理的
优点:增加了程序的可靠性,减小了memory leak
缺点:无法控制GC的时间,耗费系统性能

GC

如何确定垃圾

引用计数算法

确定哪些对象已经变成了垃圾,最简单的算法就是引用计数法
给对象添加一个引用计数器
每当一个地方引用它时,计数器加1
每当引用失效时,计数器减少1
当计数器的数值为0时,也就是对象无法被引用时,表明对象不可在使用

但是这个算法存在一个致命的缺陷,无法解决循环引用的问题

为此,引入了另外一种根搜索算法。

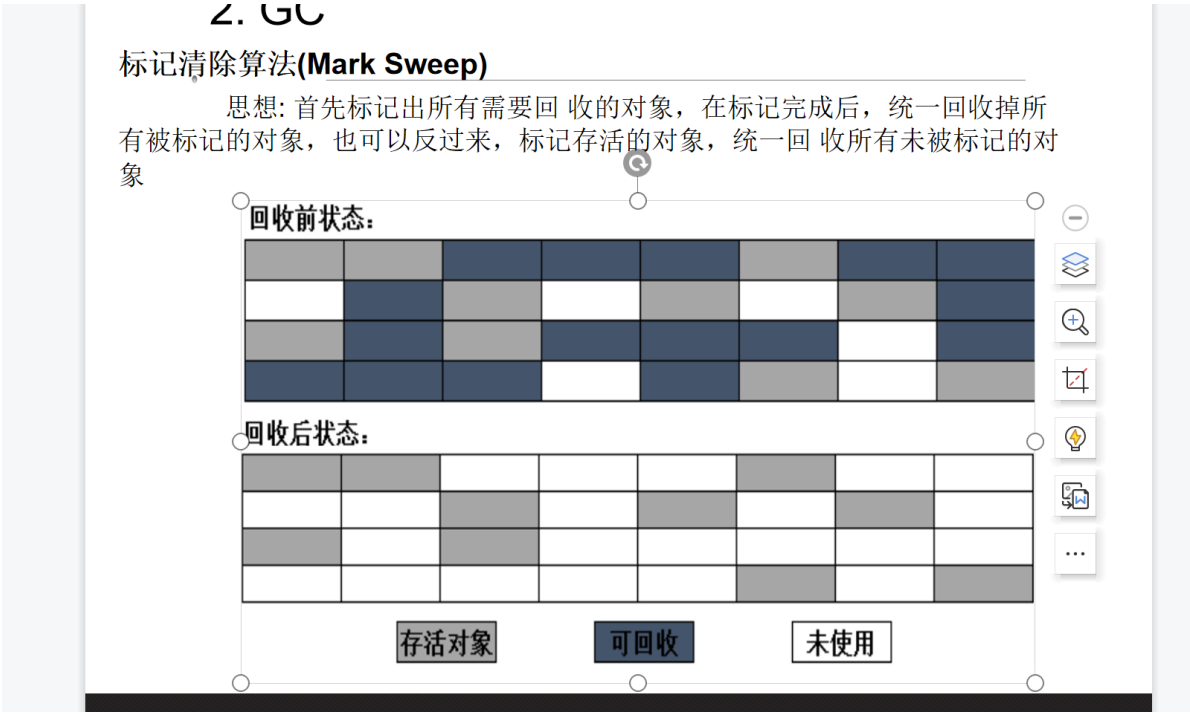
根搜索算法

这个算法的基本思想是将一系列称为“GC Roots”的对象作为起始点
从这些节点开始向下搜索
搜索所走的路径称为引用链
当一个对象到所有的GC root之间没有任何引用链相连，时，就认为该对象变成了垃圾

GC Roots包含对象呢?
虚拟机栈中引用的对象
方法区中的静态属性引用的对象

如何回收垃圾

标记清除算法



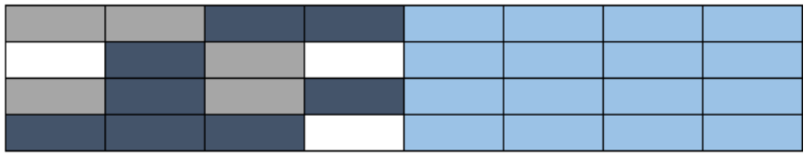
1	垃圾对象	1								
2	存活对象	0			标记清除算法					
3	未使用	-1		回收前				回收后		
4			1	0	1		-1	0	-1	
5	特点：实现简单，但是会造成垃圾碎片		0	-1	0	→	0	-1	0	假设我要创建一个数组，但是无法创建一个连续的存储空间
6			1	0	1		-1	0	-1	
7										
8										

标记复制算法

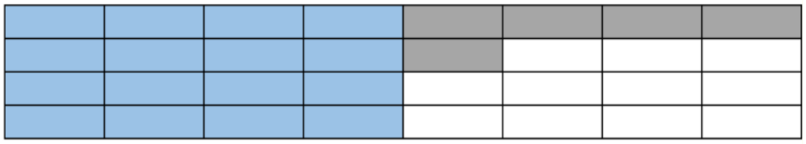
复制算法(Copy)

思想: 它将可用内存按容量划分为大小相等的两块，每次只使用其中的一块。当这一块的内存用完了，就将还存活着 的对象复制到另外一块上面，然后再把已使用过的内存空间一次清理掉。如果内存中多数对象都是存活的，这种算法将会产生大量的内存间复制的开销，但对于多数对象都是可回收的情况，算法需要复制的就是占少数的存活对象，而且每次都是针对整个半区进行内存回收，分配内存时也就不用考虑有空间碎片的复杂情况.

回收前状态：



回收后状态：



存活对象　可回收　未使用　保留区域

分配内存区域　　标记复制算法　　灰色为保留区域

回收前	1	0	1	0	0	0	特点：1.一次性回收大片的连续空间。2.不会产生内存碎片。3.如果说有少量存活对象，复制起来很快，适用于少量对象存活的情况。4.但是，内存利用率低。5.如果有大量对象存活，就需要复制大量对象，效率就低了。
	0	-1	0	0			
	1	0	1				

成为新的保留区域　　新的内存分配区

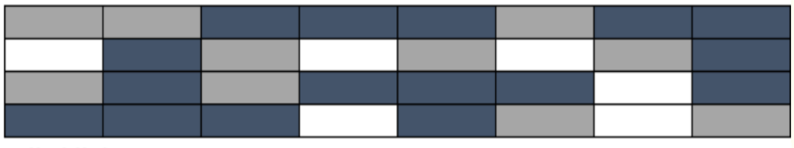
回收后			0	0	0
			0		

标记整理算法

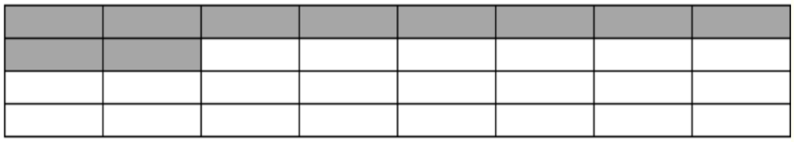
标记整理算法(Mark Compact)

思想: 其中的标记过程仍然与“标记-清除”算法一样，但后续步骤不是直接对可回收对象进行清理，而是让所有存活的对象都向内存空间一端移动，然后直接清理掉边界以外的内存

回收前状态：



回收后状态：

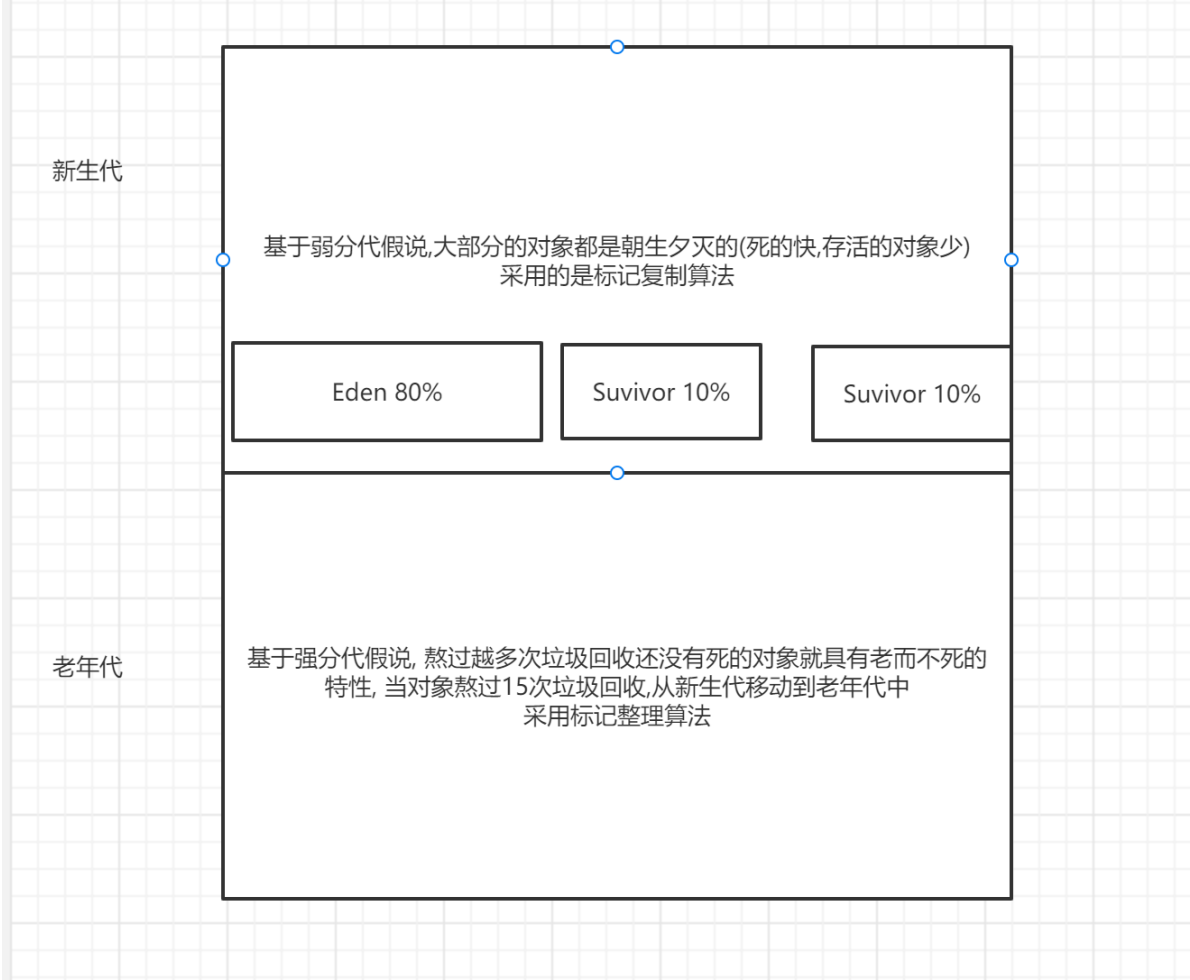


存活对象　可回收　未使用

分代收集算法

2个假说

- 强分代假说
- 弱分代假说



什么时候回收垃圾

申请heap space失败后会触发GC回收

系统进入idle后一段时间会进行回收

主动调用GC进行回收