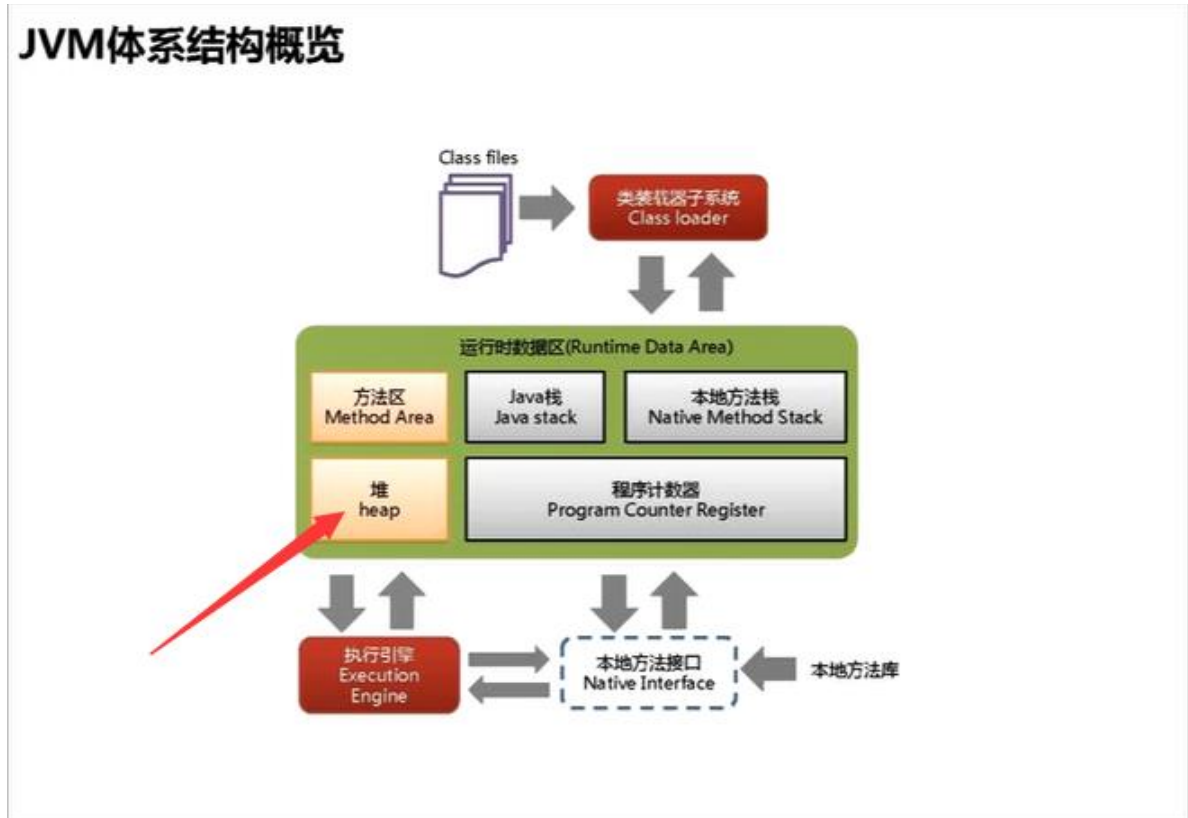


尚硅谷面试第一季-16 JVM 垃圾回收机制

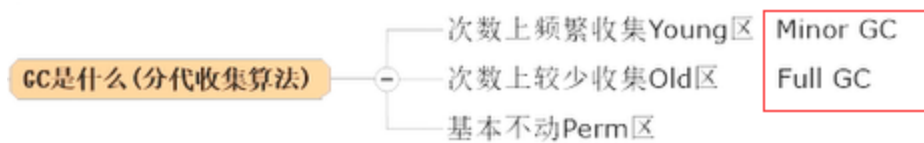
问题的提出：GC 发生在 JVM 那个部分，由几种 GC，它们的算法是什么？

课堂重点：

GC 发生在 JVM 体系的堆部分。



什么是 GC（分代手机算法）Minor GC 和 Full GC



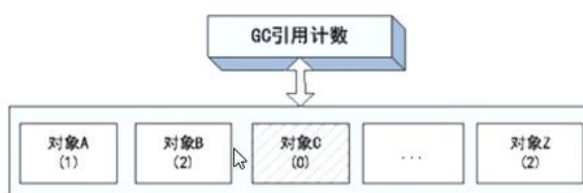
GC4 大算法：



1、引用计数法

1. 引用计数法

(应用：微软的COM/ActionScript/Python...)



缺点：

- 每次对对象赋值时均要维护引用计数器，且计数器本身也有一定的消耗；
- 较难处理循环引用

JVM的实现一般不采用这种方式

2、复制算法（Copying）

年轻代中使用的是 Minor GC，这种 GC 算法使用的是复制算法（Copying）

原理：

- 从根集合(GC Root)开始，通过Tracing从From中找到存活对象，拷贝到To中；
- From、To交换身份，下次内存分配从To开始；



3、标记清除（Mark-Sweep）

老年代一般是由标记清除或者是标记整理的混合实现

1. 标记 (Mark):

从根集合开始扫描，对存活的对象进行标记。



2. 清除 (Sweep):

扫描整个内存空间，回收未被标记的对象，使用free-list记录空闲区域。



✓ 不需要额外空间

✗ 两次扫描，耗时严重；
✗ 会产生**内存碎片**

4、标记压缩（Mark-Compact）

老年代一般是由标记清除或者是标记整理的混合实现

原理：

1. 标记 (Mark):

与 标记-清除 一样。



2. 压缩 (Compact):

再次扫描，并往一端 滑动 存活对象。



✓ 没有内存碎片，可以利用bump
-the-pointer ✗ 需要移动对象的成本

在整理压缩阶段，不再对标记的对象做回收，而是通过所有存活对象都向一端移动，然后直接清除边界以外的内存

5、标记清除压缩（Mark-Sweep-Compact）

标记-清除-压缩 (Mark-Sweep-Compact)

原理：

1. Mark-Sweep 和 Mark-Compact的结合。
2. 和Mark-Sweep一致，当进行多次GC后才Compact。



减少移动对象的成本