1、关键字说明：

1.1、GC Roots

找到“GC Roots”也是要花很长的时间，然而这里又有新的解决方法，就是通过采用一个OopMap的数据结构来记录系统中存活的“GC Roots”，在类加载完成的时候，虚拟机就把对象内什么偏移量上是什么类型的数据计算出来保存在OopMap，通过解释OopMap就可以找到堆中的对象，这些对象就是GC Roots。而不需要一个一个的去判断某个内存位置的值是不是引用。这种方式也叫准确式GC。

1.2、Stop the World

GC停顿，将非GC线程相关的用户线程临时挂起，实现对象状态的一致性，不能存在GC过程中，对象的状态飘忽不定的状态，做到准确GC。

1.3、Safe Point

程序并非在所有地方都能停下来开始GC，只有到达某个特定的位置才可以暂停，这个特定的位置就称为Safe Point。抢断式中断，首先爸所有线程全部中断，如果有线程并不处于安全点，那就恢复这个线程，让它跑到安全点上---抢先式中断；不直接对线程操作，使用标志信息，各个线程论询这个标志，发现标志为真是就挂起。轮询标志的地方和安全点是重合的—主动式中断。

1.4、Safe Region

被扩展的Safe Point，在这个区域内被认为是引用关系是不改变的，可以随时进行GC。

2、常见的垃圾回收算法

2.1、标记-清除法

是最基础的算法，后续的算法都是在此基础上不断完善和改进完成的。这个算法主要有两个不足：1、效率问题，不管是标记还是清除，效率都不高；2、空间问题，标记清除之后，会留下很多不连续的内存碎片，而空间碎片的增多不利于大对象的分配，可能会提前触发一次GC。

2.2、复制算法

将内存空间划分为大小想相等的两块，每次仅对其中的一块进行操作，当其中的一块使用完了，就将存活的对象复制到另一块上，随后将使用过的那块一次性清除干净。

这样看，一次只操作一块内存，将近50%的空间，显得太浪费了。其实，研究表明，新生代中的对象98%的属于朝生夕死的对象，所以并不需要按照1:1的形式来划分，而是按照这个比例进行划分：8:1:1的形式；

两个问题：1、为什么两块survivor区？2、survivor满了怎么办（分配担保）？

2.3、标记-整理法

标记整理算法的“标记”过程和标记-清除算法一致，只是后面并不是直接对可回收对象进行整理，而是让所有存活的对象都向一段移动，然后直接清理掉端边界意外的内存。

3、垃圾收集器

前面对几种常见的垃圾收集算法进行了简单的说明，下面所要讲到的垃圾收集器其实可看做是垃圾收集算法的具体实现。

3.1、Serial 收集器

一款采用复制算法思想实现的串行收集器，是一款发展较久的基本的收集器，一般用于新生代的垃圾收集；