1.谈谈GC

java中的GC主要是解决堆内存的分配问题。堆内存主要是存储对象实例。为了进行高效的垃圾回收，将堆内存区域分为新生代、老年代、永久代三个区域。

新生代：由Eden（e等）和两个suvivor区组成，在大多数情况下，对象在Eden的区域进行分配，当Eden没有足够的空间时，会触发一次minor GC，然后新生代就会把存活的对象移动到S0区域，并清空Eden区域，当再次发生minor GC时，将Eden和S0中存活的对象移动到S1中。
当GC后在新生代中放不下，或者对象的年龄过大，就会放入老年代中。

老年代：当老年代的空间不足时，会引发Major GC或者Full GC，速度要比minor GC慢很多

永久代：在JDK8之前，类的元数据信息（用来描述数据的数据，更通俗一点，就是描述代码间关系，或者代码与其他资源（例如数据库表）之间内在联系的数据）全部保存在永久代中，一旦元数据超过了永久代的大小，就会抛出异常。JDK8去除了永久代，将元数据信息直接保存在本地内存中。

2.垃圾收集算法。

标记-清除算法：对待回收的对象进行标记

缺点：标记和清除效率都很低，并且会产生大量的内存碎片。

复制算法：将内存划分为大小相同的A和B，当A的内存用完了，就把存活的对象复制到B中，并且清空A的内存。

优缺点：提高了效率，并且避免了内存碎片的问题，但是内存变成了原来的一半。

标记-整理算法：在老年代中，对象的存活率较高，复制算法效率比较低。在标记-整理算法中，标记出所有存活的对象，并且把它移到一端，然后直接清理边界以外的内存。

3.判断对象是否存活

引用计数法：对每一个对象加上一个引用计数器，每当一个对象引用它时，计数器加1，当使用完该对象时，计数器减1，计数器的值为0的对象表示不可能再使用。

优缺点：简单高效，但是不能解决对象相互引用的问题。

可达性分析法：当一个对象到达GC ROOT没有任何的引用链时，意味着该对象可以被回收。

可以作为GC root的对象：

本地变量表中引用的对象

方法区中静态变量引用的对象

方法区中常量引用的对象

Native方法引用的对象