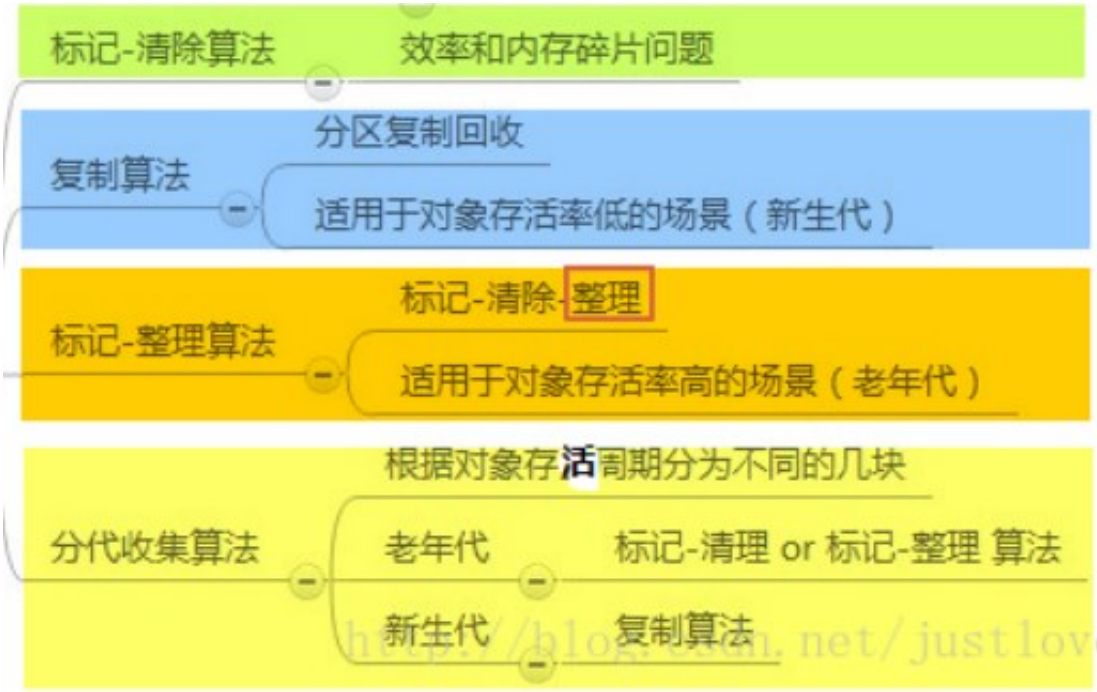
1、谈一下你对垃圾回收机制的理解：
垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清除和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。

- **那些内存需要回收？**(对象是否可以被回收的两种经典算法: 引用计数法 和 可达性分析算法)
- 引用计数法：**任何引用计数为0的对象实例可以被当作垃圾收集。**
- **可达性分析算法：通过判断对象的引用链是否可达来决定对象是否可以被回收**
- **什么时候回收？**（堆的新生代、老年代、永久代的垃圾回收时机，MinorGC 和 FullGC）
- **新生代:尽可能快速的收集掉那些生命周期短的对象，一般情况下，所有新生成的对象首先都是放在新生代的**
- **老年代:存放的都是一些生命周期较长的对象,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**
- **永久代:主要用于存放静态文件，如Java类、方法等**

由于对象进行了分代处理，因此垃圾回收区域、时间也不一样。**垃圾回收有两种类型，Minor GC 和 Full GC。**

- **Minor GC：**对新生代进行回收，不会影响到年老代。因为新生代的 Java 对象大多死亡频繁，所以 Minor GC 非常频繁，一般在这里使用速度快、效率高的算法，使垃圾回收能尽快完成。
- **Full GC：**也叫 Major GC，对整个堆进行回收，包括新生代、老年代和永久代。由于 Full GC需要对整个堆进行回收，所以比Minor GC要慢，因此应该尽可能减少Full GC 的次数，导致Full GC的原因包括：老年代被写满、永久代（Perm）被写满和System.gc()被显式调用等
-
- **如何回收？**(三种经典垃圾回收算法(标记清除算法、复制算法、标记整理算法)及分代收集算法 和 七种垃圾收集器)


标记-清除算法
效率和内存碎片问题
复制算法
分区复制回收
适用于对象存活率低的场景（新生代）
标记-整理算法
标记-清除-整理
适用于对象存活率高的场景（老年代）
分代收集算法
根据对象存活周期分为不同的几块
老年代
标记-清理 or 标记-整理 算法
新生代
复制算法