Java 语言中一个显著的特点就是引入了垃圾回收机制，这个大家都清楚，垃圾回收的概念这里也不做介绍，重点是垃圾回收是在什么时候开始？对什么东西，做了什么事情？

GC 何时开始：

所有的回收器类型都是基于分代技术来实现的，那就必须要清楚对象按其生命周期是如何划分的。

- 年轻代：划分为三个区域：原始区(Eden)和两个小的存活区(Survivor)，两个存活区按功能分为 From 和 To。绝大多数的对象都在原始区分配，超过一个垃圾回收操作仍然存活的对象放到存活区。垃圾回收绝大部分发生在年轻代。
- 年老代：存储年轻代中经过多个回收周期仍然存活的对象，对于一些大的内存分配，也可能直接分配到永久代。
- 持久代：存储类、方法以及它们的描述信息，这里基本不产生垃圾回收。

有了以上这些铺垫之后开始回答 GC 何时开始：

Eden 内存满了之后，开始 Minor GC（从年轻代空间回收内存被称为 Minor GC）；升到老年代的对象所需空间大于老年代剩余空间时开始 Full GC（但也可能小于剩余空间时，被 HandlePromotionFailure 参数强制 Full GC）

对什么东西操作，即垃圾回收的对象是什么：

从 root 开始搜索没有可达对象，而且经过第一次标记、清理后，仍然没有复活的对象。

做了什么东西：

主要做了清理对象，整理内存的工作。具体的引申如下

垃圾回收器的类型：

- 串行垃圾回收器（Serial Garbage Collector）
- 并行垃圾回收器（Parallel Garbage Collector）
- 并发标记扫描垃圾回收器（CMS Garbage Collector）
- G1 垃圾回收器（G1 Garbage Collector）

垃圾回收算法：

- 引用计数法
- 标记清除法
- 复制算法
- 标记压缩算法
- 分代算法
- 分区算法

类在虚拟机中的加载过程

加载 Loading：
通过一个类的全限定名来获取一个二进制字节流、将这个字节流所代表的静态存储结构转化为方法区的运行时数据结构、在内存中生成一个代表这个类的 java.lang.Class 对象，作为方法区这个类的各种数据的访问入口。
验证 Verification：
确保 Class 文件的字节流中包含的信息符合当前虚拟机的要求，并不会危害虚拟机的自身安全。
准备 Preparation：
正式为类变量分配内存并设置类变量初始值。
解析 Resolution：
虚拟机将常量池内的符号引用替换为直接引用的过程。
初始化 Initialization：
类加载过程的最后一步，到了这个阶段才真正开始执行类中定义的 Java 程序代码。
使用 Using：
根据你写的程序代码定义的行为执行。
卸载 Unloading：
GC 负责卸载，这部分一般不用讨论。

强引用、软引用、弱引用、虚引用与 GC 的关系

强引用：new 出的对象之类的引用，只要强引用还在，永远不会回收。
软引用：引用但非必须的对象，内存溢出异常之前回收。
弱引用：非必须的对象，对象只能生存到下一次垃圾收集发生之前。
虚引用：对生存时间无影响，在垃圾回收时得到通知。