http://www.jianshu.com/p/0ef07636796e

GC(Garbage Collection)机制，是Java与C++/C的主要区别之一,Java开发者，一般不需要单独处理内存的回收，GC会负责内存的释放。
java运行时区域中程**序计数器、虚拟机栈、本地方法栈**3个区域随线程生命周期结束而结束，**Java堆、方法区**则不一样，一个接口中的多个实现类需要的内存可能不一样，一个方法中的多个分支需要的内存也可能不一样，我们只有在程序处于运行期间时才能知道会创建哪些对象，这部分内存的分配和回收都是动态的，垃圾收集器所关注的是这部分内存。

一.对象是否死亡

判断对象是否死亡通常有**引用计数算法、可达性分析算法**

引用计数算法

该算法通常是给对象中添加一个引用计数器，每当有一个地方引用它是，计数器就会+1，引用失效，计数器值就会-1，如果计数器值为0，对象就不会被再使用，就可以回收该对象了。
但该算法很难解决**循环引用**的问题。

可达性分析算法

在主流的商用程序语言（Java、C#）的主流实现中，都是称通过可达性分析（Reachability Analysis）来判定对象是否存活的。
基本思路就是通过一系列的称为“GC Roots”的对象作为起点，从这些节点开始向下搜索，搜索所走过的路径称为**引用链（Reference Chain）**，当一个对象到GC Roots没有任何引用链相连（用图论的话来说，就是从GC Roots到这个对象不可达）时，则证明此对象是不可用的。
GC Roots对象包括以下几种:

- 虚拟机栈（栈帧中的本地变量表）中引用的对象
- 方法区中类静态属性引用的对象
- 方法区中常量引用的对象
- 本地方法栈JNI（即一般说的Native方法）引用的对象

引用

判定对象是否存活都与"引用"有关,Java对引用概念进行了扩充。

强引用（Strong Reference）

软引用（Soft Reference）

弱引用（Weak Reference）

虚引用（Phantom Reference）

二.垃圾收集算法

1.标记-清除算法

首先标记出所有需要回收的对象，再标记完成后统一回收所有被标记的对象。效率不高，且容易产生内存碎片。

2.复制算法

为了解决效率问题，“复制”（Copying）的收集算法将可用内存按容量划分为大小相等的两块，每次只使用其中一块。当这一块的内存用完了，就将还存活着的对象复制到另一块上面，然后再把已使用过的内存空间一次清理掉。

一般是将内存分为一块较大的Eden空间和两块较小的Survivor空间，每次使用Eden和其中一块Survivor。当回收时，将Eden和Survivor中还存活着的对象一次性地复制到另一块Survivor空间上，最后清理掉Eden和刚才用过的Survivor空间。

3.标记-整理算法

标记过程仍然与“标记-清除”算法一样，但后续步骤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端移动，然后直接清理掉端边界以外的内存。

4.分代收集算法

当前商业虚拟机的垃圾收集都采用“分代收集”算法，一般把Java堆分为**新生代**和**老生代**，这样就可以根据各个年代的特点采用最合适的收集算法。

三.垃圾收集器

Serial收集器

ParNew收集器

Parallel Scavenge收集器

Serial Old 收集器

Parallel Old收集器

CMS收集器

G1收集器

一般收集器都需要“Stop The World”

四.内存分配与回收策略

- 对象优先在Eden分配
- 大对象直接进入老年代
- 长期存活的对象将进入老年代
- 动态对象年龄判定
- 空间分配担保