java的内存自动管理主要分为两种：1.内存的分配 2内存的回收
对象的内存分配往大的方向去说，就是在堆上分配，对象主要分配在新生代的Eden区中，如果启动了本地线程分配缓冲区，将按线程优先在TLAB上分配，少数也会直接分配到老年代的

1.对象优先分配到Eden区中
对象优先在新生代的Eden去中进行分配，当Eden区中没有足够的空间进行分配时，就会发生MinorGc
使用-XX:+PrintGCDetails可以查看在虚拟机发生垃圾收集行为时打印内存回收情况， 并且在线程退出的时候输出当前内存各个区域的分配情况 -Xms20M即分配堆上内存最大为20M -Xmx20M 即最小内存20M -Xmn10M 新生代大小10M -XX:SurvivorRatio=8决定了新生代中Eden区与另一个Survivor区的空间比例为8:1， 从输出情况可以看到 eden space 8192k， from space 1024k, to space 1024k，新生代的总可用空间为9216kb(Eden区加上一个Survivor区的总容量)

minor gc 和 fullgc的区别：
1. 新生代gc（minorgc）:指发生在新生代的垃圾回收动作，因此java对象大多数都是具备朝生夕死的特征，所以MinorGC很频繁，一般回收速度也比较快
2. 老年代GC（MajorGC/Full GC）： 指老年代发生的GC，出现了Major GC经常伴随至少一次Minor Gc（并非绝对， Parallel Scavenge收集器的收集策略里就有直接进行MajorGC的策略选择），MajorGC的速度一般会比Minor gc慢上10倍以上

2. 大对象会直接进入老年代
所谓大对象是指，占用连续空间的对象，最典型的就是字符串以及数组，大对象的分配对虚拟机的内存分配就是一个坏消息，比大对象还要麻烦的就是遇到短命的大对象，写程序要尽量避免。经常出现大对象容易导致内存还有不少空间时就要提前触发垃圾回收来获取连续的空间进行安置
虚拟机提供-XX:PertenureSizeThreshold参数，当对象大于这个参数的时候，直接分配到老年代，避免在Eden区以及两个Survior区中发生大量的复制，这个参数只对Serial 和 ParNew有效

3. 长期存活的对象进入老年代

当对象发生一次Minor gc仍然存活并且能被Survior区容纳的话，年龄就会增加1，当对象的年龄熬到一定年龄 （默认15）的时候，就会进入老年代。进入老年代的年龄的阈值可以通过-XX:MaxTenuringThreshold=1来设置

4. 对象年龄动态判断
不是说所有的对象年龄必须达到了MaxTenuringThreshold才会进入老年代的，如果在Survivor空间内，相同年龄所有对象的总和占了Survivor空间的一半，年龄大于或者等于该年龄的对象可以直接进入老年代，无须等待MaxTenuringThreshold

5. 空间分配担保
在发生MinorGC之前，虚拟机会检查老年代的最大可用连续内存是否大于新生代所有对象空间，如果这个条件成立，那么MinorGC是安全的，如果不满足，就会查看是否开启了分配担保（即HandlePromotionFailure），如果允许，那么就会继续检查老年代的最大可用连续空间是否大于晋升到老年代对象的平均大小，如果大于就会尝试进行一次MinorGC，尽管这个MinorGC是有风险的，如果小于或者HandlePromotionFaulure设置不允许冒险，那这时也要改为进行一次Fullgc
为什么说进行分配担保是有风险的，因为进行分配担保之前会进行内存大小检查， 可能检查的时候内存大小是满足的，但是当分配的时候，发现内存不足，这时就要再进行Full gc，会绕一圈，耽误更多时间。在JDK6以后，这个参数不在使用，规整调整为只要老年代的连续空间大于新生代对象总大小或者历次晋升的平均大小就会进行MinorGC，否则进行Full gc

jdk1.8默认使用的垃圾回收器是 Parallel Scavenge + Parallel Old，因此使用以上参数的时候，需要调整为Serial 或 ParNew收集器
-XX:+PrintCommandLineFlagsjvm参数可查看默认设置收集器的类型