# JVM相关知识点补充

主要是我看《深入理解Java虚拟机》的一些笔记。

## 内存分配与回收策略

1. 对象优先在Eden区分配，如果Eden区空间不够，则发起Minor GC。
2. 大对象直接进老年代。大对象是很讨厌的，比大对象更讨厌的是用完就丢的大对象。Eden区小，一个大对象可能就把Eden区占满，触发Minor GC。直接放入老年代就相对好一些，但是如果几个大对象就把老年代占满了，又要触发Full GC，太浪费性能了。
3. 长期存活的对象进老年代。在Eden区熬过一次Minor GC并能放入Survivor区的话，这个对象年龄为1，然后在Survivor区熬过一次Minor GC，年龄就+1。到15岁（这个阈值可配置）时，进入老年代。
4. 动态对象年龄判断。假如Survivor区内相同年龄的所有对象大小总和大于Survivor区大小的50%，则年龄大于或等于该年龄的对象就直接进入老年代，不用非得等到上面配置好的年龄阈值（默认15）。有点机构人员冗余，年龄偏大的人员提前退休的意思。
5. 空间分配担保。在Minor GC之前，JVM要做一些检查：

老年代连续空间是否大于新生代现在所有对象空间总和？

是：Minor GC是安全的。因为在极端情况下，新生代所有对象都可以放进老年代。

否：查一下HandlePromotionFailure是否允许担保失败。

否：先执行Full GC，老年代腾出空间，以保证极端情况下，Minor GC之后新生代所有对象都能进入老年代。

是：老年代最大连续空间是否大于历次晋升老年代对象的平均大小？

否：执行Full GC。

是：执行Minor GC。有风险，因为晋升老年代的对象有可能大于平均值。