前言

本节的内容是继续深化JVM的空间分配。总体上JVM的内存分配需要建立的知识架构是：

（1）知道JVM内存为基本的三块儿：堆，方法区，方法栈。掌握调节三块内存空间的命令。

（2）熟悉堆内存和栈内存的特点（存放内容，回收方式），熟悉常量池

（3）继续深化堆的认识【本节课知识所涉及的内容】

（4）内存相关的命令【本节课知识所涉及的内容】

## 垃圾回收中的有向图原理[理解]

将对象考虑为有向图的顶点；将引用关系考虑为图的有向边，并且从引用者指向被引用者；

每个线程对象作为图的起始顶点，通常为main线程。以下面的代码和其对应的内存分布图来说明垃圾回收的过程：

class test

{

public void mian(String args[])

{

Object o1 = new Object();//➀

Object o2 = new Object();//➁

o2=o1;//➂

}

}



当程序执行到➂的时候，垃圾回收器就可以回收堆内存中的object2了。

## 内存回收的基本算法[理解]

第一：复制

将堆内存分成两个相同空间，从根（有向图的起始点）开始访问每一个关联的可达对象，将空间A的可达对象全部复制到空间B，然后一次性回收整个空间A。

只需要访问所有可达对象，将所有可达对象复制走以后，就可以回收整个空间，不需要访问那些不可达的对象，所以遍历成本较小。但是需要较大的复制成本和较多的内存

第二：标记清除

垃圾回收器先从根开始访问所有的可达对象，将它们标记为可达状态，然后再遍历一次整个内存区域，把所有没有标记为可达的对象进行回收处理。

无需大规模的复制操作，而且内存利用率高。但是需要两次遍历堆内存空间，遍历的成本较大，造成程序暂停的时间随堆空间大小线性增长。收回的内存往往不是连续的，堆内存的碎片多。

第三：标记压缩

充分利用上述两种算法的优点，垃圾回收器先从根开始访问所有可达的对象，将它们标记为可达状态，接下来垃圾回收器会将这些对象搬迁到一起，这个过程也称内存压缩，然后垃圾回收机制再次回收那些不可达对象所占的内存空间，这样就避免了回收产生的内存碎片。

## 堆的分代思想[掌握]：

当代商业虚拟机都采用分代收集的算法。一般将Java堆分为新生代和老年代。这样就可以根据各个年代的特点采用最适当的收集算法了。在新生代中，每次垃圾收集都会发现有大批的对象死去，只有少数存活，那就采用复制算法。而老年代中因为对象存活率高，没有额外的空间进行分配担保，就必须采用“标记-清理/压缩”的算法来进行回收。

复制算法的原理是将可用的内存分成大小相等的两块，每次只使用其中的一块儿。当这一块儿内存用完之后，就将还活着的对象复制到另外一块儿上面，然后再把已经使用过的内存空间一次性的清理掉。研究表明：新生代的对象98%都是朝生夕死的，所以并不需要按照1:1的比例来划分内存空间，而是将内存划分为一块较大的eden和两块较小的survivor空间。每次使用eden和其中的一块survivor。当回收的时候，将eden和survivor中还存活的对象一次性的拷贝到另外的一块儿survivor空间，最后清理掉eden和刚才使用过的survivor空间。

-Xmn：新生代的内存空间大小。注意：此处的大小是（eden+ 2 survivor space)。整个堆大小=新生代大小+老生代大小。在保证堆大小不变的情况下，增大新生代后，将会减小老生代大小。此值对系统性能影响较大，Sun官方推荐配置为整个堆的3/8。

-XX:SurvivorRatio：新生代中Eden区域与Survivor区域的容量比值，默认值为8。两个Survivor区与一个Eden区的比值为2:8，一个Survivor区占整个年轻代的1/10。

-XX:NewRatio=xx，新生代和老年代的比例设置。新生代（包括Eden和两个Survivor区）与年老年代的比值。例如：设置为4，则新生代与老年代所占比值为1：4，新生代占整个堆栈的1/5【此处了解即可，不要求掌握】

## 分析内存常用的命令[了解]

（1）打印GC信息：

-XX:+PrintGCDetails //打印GC详细信息到控制台

-XX:+PrintGCDetails -Xloggc:gc.log//打印到gc.log文件中

（2）输出堆快照：

-XX:+HeapDumpOnOutOfMemoryError

-XX:HeapDumpPath=E:\Java\dump

JVM会在遇到OutOfMemoryError时拍摄一个“堆转储快照”，并将其保存在一个文件中。文件存放的目录为：E:\Java\dump。-XX:HeapDumpPath=${目录}参数表示生成dump文件的路径，也可以指定文件名称，例如：-XX:HeapDumpPath=${目录}/java\_heapdump.hprof。如果不指定文件名，默认为：java\_<pid>\_<date>\_<time>\_heapDump.hprof。

**最后，JVM内存研究到此结束。不建议你在这块儿花费太多的时间，因为JVM的型号太多，每个JVM都有自己的特点，很多细节知识即使掌握了用处也不大。不要觉得JVM研究的透彻就是高手，更多的时候是钻入了牛角尖了。**