* 代理模式

1. 静态代理
2. Jdk动态代理
3. cglib动态代理

* 引用

1. 强引用
2. 软引用
3. 弱引用
4. 虚引用

* 自定义ClassLoader加载class文件
* ASM/javaassist动态生成字节码

垃圾回收

1. 如何确定某个对象是垃圾
   1. 引用计数法
   2. 可达性分析法
2. 垃圾回收算法

2.1 标记-清除算法(Mark-Sweep)

2.2 复制算法(Coping)

2.3 标记-整理算法(Mark-Compact)

2.4 分代收集算法(Generational Collection)

1. 垃圾收集器

3.1 Serial/Serial Old

3.2 ParNew

3.3 Parallel Scavenge

3.4 Parallel Old

3.5 CMS

3.6 G1

1. 在发生垃圾收集行为时打印内存回收日志，并且在进程退出的时候输出当前的内存各区域分配情况

-XX:+PrintGCDetails

1. 设置threshold让大对象直接进入老年代

-XX:PretenureSizeThreshold

1．分析如下设置后打印的当前内存各区域分配情况

-Xms10m -Xmx10m -Xmn5M -XX:SurvivorRatio=8 -XX:+PrintGCDetails

Heap  
PSYoungGen total 4608K, used 1830K [0x00000000ffb00000, 0x0000000100000000, 0x0000000100000000)  
eden space 4096K, 44% used [0x00000000ffb00000,0x00000000ffcc99d8,0x00000000fff00000)  
from space 512K, 0% used [0x00000000fff80000,0x00000000fff80000,0x0000000100000000)  
to space 512K, 0% used [0x00000000fff00000,0x00000000fff00000,0x00000000fff80000)  
PSOldGen total 7168K, used 7168K [0x00000000ff400000, 0x00000000ffb00000, 0x00000000ffb00000)  
object space 7168K, 100% used [0x00000000ff400000,0x00000000ffb00000,0x00000000ffb00000)  
PSPermGen total 21248K, used 3260K [0x00000000fa200000, 0x00000000fb6c0000, 0x00000000ff400000)  
object space 21248K, 15% used [0x00000000fa200000,0x00000000fa52f100,0x00000000fb6c0000)

2．发现在上述设置中仅仅当-XX:SurvivorRatio=8设置成8的时候新生代/eden/from三者的大小才符合比例，设置成其他值后比例都不符合。查了一下，说是HotSpot VM里ParallelScavenge系的GC（UseParrallelGC/UseParallelOldG）默认行为是SurvivorRatio如果不显式设置就没啥用，显式设置只有跟默认值（8）一样时才会有效。否则，即便设置-XX:-UseAdaptiveSizePolicy也没用，因为ParallelScavenge系的GC最初设计就是默认打开AdaptiveSizePolicy的，它会自动/自适应地调整各种参数