# Homework 9-JVM垃圾回收原理

## JVM垃圾回收原理

### 标记回收对象

Java堆是JVM主要的内存管理区域，里面存放着大量的对象实例和数据，在垃圾回收算法和垃圾收集器之前，首先要做的是判断哪些对象已经死去，需要进行回收即不可能再被任何途径使用的对象。有两种思路可以判断对象存活，

- 引用计数法

给对象中添加一个引用计数器，每当有一个地方使用它时，计数器值就加1。当引用失效时，计数器就减1。任何时刻计数器为0的对象就是不可能再被使用的。

引用计数法的一个最大的缺点是很难解决对象之间互相循环引用的问题。

- 可达性分析法

通过一些列成为“GC Roots”的对象作为起点，从这些节点开始向下搜索，如果从GC Roots到一个对象不可达，则证明对象是不可用的，

Java语言中，可作为GC Roots的对象包括下面几种：

1. 虚拟机栈（栈帧中的本地变量表）中引用的对象
2. 本地方法栈JNI(即一般说的Native方法)引用的对象
3. 方法区中类静态常量引用的对象
4. 方法区中常量引用的对象

   对于Java程序而言，对象基本都位于堆内存中，简单来说GC Roots就是有被堆外区域引用的对象。

### 四种引用

在`JDK 1.2`以前的版本中，若一个对象不被任何变量引用，那么程序就无法再使用这个对象。也就是说，只有对象处于(`reachable`)**可达状态**，程序才能使用它。

从`JDK 1.2`版本开始，对象的引用被划分为`4`种级别，从而使程序能更加灵活地控制**对象的生命周期**。这`4`种级别**由高到低**依次为：**强引用**、**软引用**、**弱引用**和**虚引用**。

#### 强引用(StrongReference)

强引用是使用最普遍的引用，如下的方式就是强引用：

```
Object strongReference = new Object();
```

1. 如果一个对象**具有强引用**，那垃圾回收器绝**不会回收它**。直到强引用的对象不使用或者超出对象的

生命周期范围。则GC认为该对象不存在引用，这时候就可以回收这个对象。
2. 当内存不足时，JVM宁愿抛出OutOfMemoryError的错误，使程序异常终止，也不会靠随意回收具有强引用对象来解决内存不足的问题。

### 软引用(SoftReference)

如果对象**只具有软引用**，则

1. 内存空间充足时，垃圾回收器不会回收
2. 内存空间不足时，就会尝试回收这些对象。只要垃圾回收器没有回收它，该对象就可以被程序使用

```
SoftReference<String> softReference = new SoftReference<String>(str);
```

### 弱引用(WeakReference)

相比较软引用，**只具有弱引用的对象拥有更短暂的生命周期**。在垃圾回收器线程扫描它锁管辖的内存区域的过程中，一旦发现了**只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存**。不过，由于垃圾回收器是一个优先级很低的线程，因此不一定会很快发现那些只具有弱引用的对象。

```
WeakReference<String> weakReference = new WeakReference<>(str);
```

### 虚引用(PhantomReference)

虚引用顾名思义，就是形同虚设。与其他几种引用都不同，虚引用并不会决定对象的生命周期。如果一个对象**仅持有虚引用**，那么它就和没有任何引用一样，在任何时候都可能被垃圾回收器回收。

## 垃圾回收算法

### 标记-清除算法

标记-清除算法分为“标记”和“清除”两个阶段，

1. 标记：首先标记出所有需要回收的对象
2. 清除：在标记完成后统一回收所有被标记的对象

标记-清除算法主要有两个不足：

1. 效率问题，标记和清除的两个过程效率都不高
2. 标记-清除会产生大量不连续的内存碎片，这会导致在后面需要分配连续的大对象时，无法找到足够大的连续内存而导致不得不提前触发另一次垃圾收集动作

### 复制算法

复制算法的大致思路如下，

1. 首先将可用内存分为大小相等的两块，每次只使用其中的一块。
2. 当这一块的内存用完了，就将还存活的对象连续复制到另一块上面，然后把使用过的内存空间一次清理掉

复制算法的代价就是将内存缩小为原来的一半。

现在的商业虚拟机都是采用复制算法来回收新生代。

1. 新生代的内存分为一块较大的Eden空间和两块较小的Survivor空间。
2. 每次使用Eden和一块Survivor，当进行回收是，将Eden和Survivor中还存活的对象一次性复制到另一个Survivor空间上。然后，清理掉Eden和刚刚使用过的Survivor空间。
3. HotSpot虚拟机默认Eden和Survivor的大小比例为8 : 1，这样每次新生代可用内存为整个新生代的90% (10% + 80%)，只有10%的内存会被浪费。

## 标记-整理算法

标记-整理算法分为“标记”和“整理”两个阶段：

1. 标记：首先标记出所有需要回收的对象
2. 整理：**让所有的存活的对象都向一端移动，然后直接清除掉边界以外的内存**

## 分代收集算法

分代收集算法就是降Java堆分为新生代和老年代，根据其各自的特点采用最适当的收集算法。

1. 新生代中大批对象死去，只有少量存活，就选用复制算法
2. 老年代中对象存活几率高，没有额外的空间对它进行分配担保，就必须使用标记-清除或者标记-整理算法。