Java的数据类型有基本类型和引用类型，基本类型有8种，引用类型是怎么分类的呢，让我们以前研究下这引用类型的分类吧。

JDK1.2以后引用被分为四类：强引用、软引用、弱引用、虚引用。这样分类是为了更加灵活的控制对象的生命周期。

1、强引用

强引用是最普通的引用类型，日常开发中使用的引用类型默认都是强引用类型，强引用的对象只要引用还存在，GC就不会回收这个对象。JVM不会回收存在引用的强引用类型对象来解决内存不足的问题。如下代码

Object obj = new Object();

只要obj这个引用还存在，Java内存不足时，即使抛出内存溢出的异常也不会回收obj指向的对象。

试着运行如下代码

public static void main(String[] args) {  
 ArrayList list= new ArrayList();  
 while(true) {  
 String obj = System.*currentTimeMillis*() +"";  
 list.add(obj);  
 System.*out*.println(list.get(0));  
 }  
}

设置jvm参数-server -Xms5m -Xmx5m，最终会内存溢出Exception in thread "main" java.lang.OutOfMemoryError: GC overhead limit exceeded，内存回收效率过低，即内存泄漏

2、软引用

对象的引用软引用的情况下，内存充足，对象就不会被回收；内存不足，则回收这个对象。软引用一般作为高速缓存使用。

创建一个软引用，并执行一次GC试试

SoftReference<String> reference = new SoftReference("mg20200627");  
System.*gc*();  
System.*out*.println(reference.get());

Reference对象依旧会在。

试着把内存塞满，看看会有发生什么，设置jvm参数-server -Xms5m -Xmx5m，执行下代码

SoftReference<MgDemo202006> reference = new SoftReference(new MgDemo202006());ArrayList list = new ArrayList();  
 while(true)  
 {  
 String obj = System.*currentTimeMillis*() +"";  
 list.add(obj);  
 System.*out*.println(reference.get());  
 }

一段时间后，reference指向对象为null，内存不足的情况下，软应用对象被回收。

3、弱引用

弱引用的生命周期相对较短，GC过程中对象的引用只有弱引用的情况下，不管内存时候充足，对象都会被回收。

创建一个弱引用对象，然后执行一次GC，会有什么结果呢

WeakReference<MgDemo202006> reference = new WeakReference(new MgDemo202006());  
System.*out*.println(reference.get());  
System.*gc*();  
System.*out*.println(reference.get());

GC结束后reference指向对象为null，TreadLocal中的Entry使用的就是ThreadLocal的弱引用。

4、虚引用

虚引用的对象和没有引用的对象一个待遇，任何时候都有可能会被回收。

虚引用需要和引用队列结合使用，执行下面的代码

ReferenceQueue<MgDemo202006> referenceQueue = new ReferenceQueue<>();  
PhantomReference<MgDemo202006> reference = new PhantomReference(new MgDemo202006(),referenceQueue);  
System.*out*.println(reference.get());

执行结果为null，暂时没有找到虚引用的应用场景。

到这里引用的分类就结束了，有问题可以留言，或者关注公众号 MG驿站。