## 一. 强引用

　1. 概念

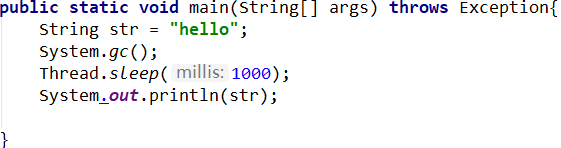
用的最广。我们平时写代码时，new一个Object存放在堆内存，然后用一个引用指向它，这就是强引用。

　2. 何时回收

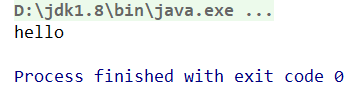
**如果一个对象具有强引用，那垃圾回收器绝不会回收它**。当内存空间不足，Java虚拟机宁愿抛出OutOfMemoryError错误，使程序异常终止，也不会靠随意回收具有强引用的对象来解决内存不足的问题。

　3. 代码示例

（1）代码



（2）运行结果



## 二. 软引用

　1. 概念

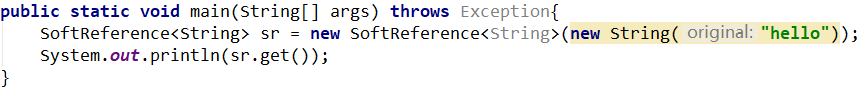
　 软引用是用来描述一些有用但并不是必需的对象，在Java中用java.lang.ref.SoftReference类来表示。

　2. 何时回收

如果一个对象只具有软引用，则内存空间足够时，垃圾回收器就不会回收它；如果**内存空间不足了，就会回收这些对象的内存**。（备注：如果内存不足，随时有可能被回收。）

　　只要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。

　3. 代码示例



## 三. 弱引用

　1. 概念

　 弱引用也是用来描述非必需对象的，在java中，用java.lang.ref.WeakReference类来表示。

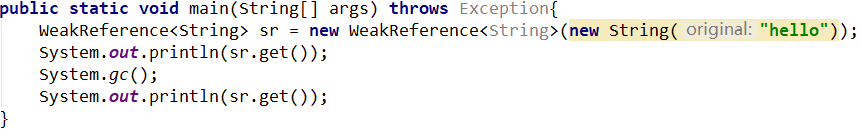
　2. 何时回收

弱引用与软引用的区别在于：**只具有弱引用的对象拥有更短暂的生命周期**。

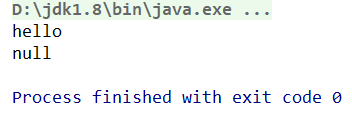
**每次执行GC的时候**，一旦发现了只具有弱引用的对象，**不管当前内存空间足够与否，都会回收它的内存**。不过，由于垃圾回收器是一个优先级很低的线程，因此**不一定会很快发现那些只具有弱引用的对象**。

　3. 代码示例

（1）代码



（2）运行结果



## 四. 虚引用

　1. 概念

虚引用和前面的软引用、弱引用不同，它并不影响对象的生命周期。在java中用java.lang.ref.PhantomReference类表示。如果一个对象与虚引用关联，则跟没有引用与之关联一样，在任何时候都可能被垃圾回收器回收。

　2. 何时回收

“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**。

　　虚引用主要用来跟踪对象被垃圾回收器回收的活动。

　3. 代码示例

要注意的是，虚引用必须和引用队列关联使用，当垃圾回收器准备回收一个对象时，如果发现它还有虚引用，就会把这个虚引用加入到与之 关联的引用队列中。程序可以通过判断引用队列中是否已经加入了虚引用，来了解被引用的对象是否将要被垃圾回收。如果程序发现某个虚引用已经被加入到引用队列，那么就可以在所引用的对象的内存被回收之前采取必要的行动。

（1）代码

