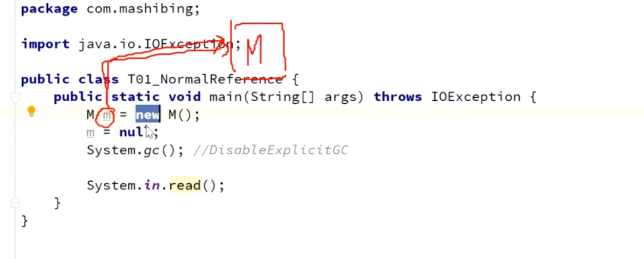
1. 强引用



定义：直接指向堆中对象的引用

GC时间：当没有任何引用指向堆中对象时

场景：程序员需要使用的

1. 软引用

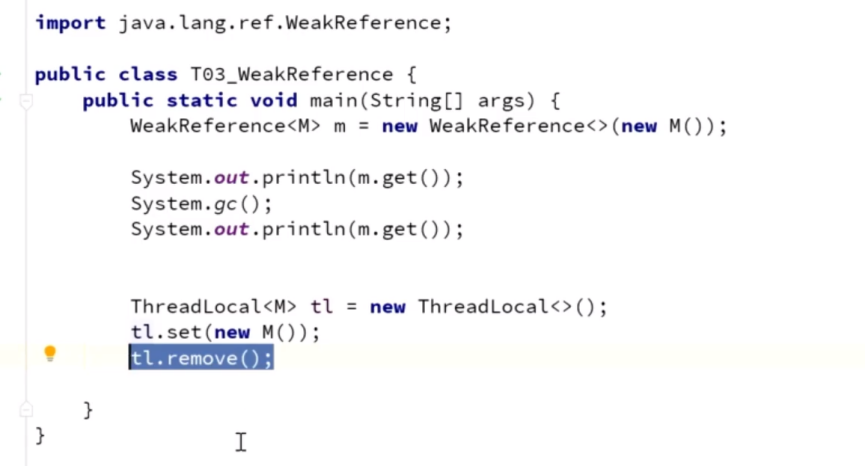


定义：不直接指向堆中对象，通过SofeReference指向堆中对象的引用

GC时间：当内存空间足够的时候，不会被回收；当内存空间不够时，会被回收

场景：缓存，比如：加载图片到内存，图片很占内存。若使用强引用，不用的时候直接GC，需要的时候需要再次加载，这样就效率很低；但是可以通过软引用指向它，使用的时候缓存下来，不使用的时候内存不够才GC，兼顾时间和空间的效率。

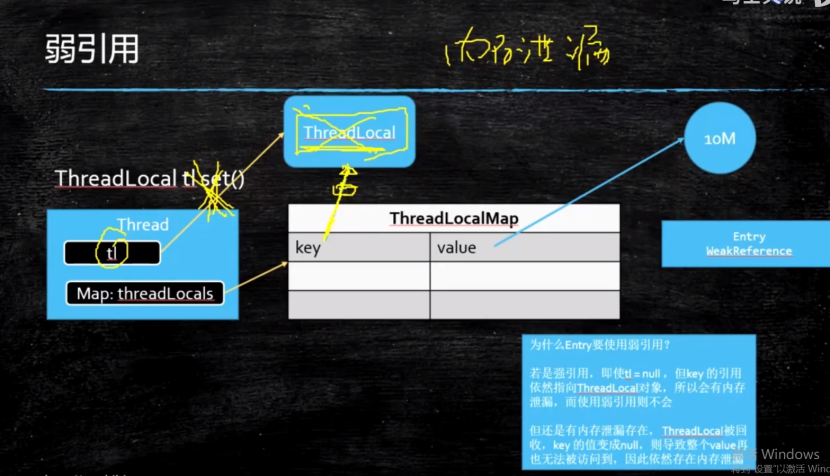
1. 弱引用



定义：不直接指向堆中对象，通过WeakReference指向堆中对象的引用

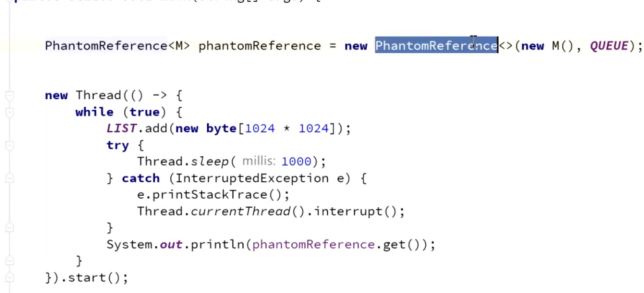
GC时间：执行System.gc()

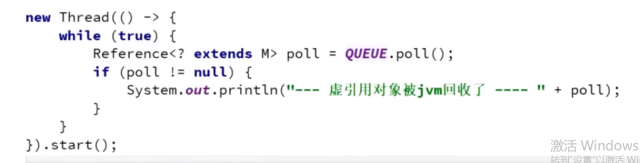
场景：防止内存泄漏，若key指向的ThreadLocal是一个强引用，则很可能会内存泄漏

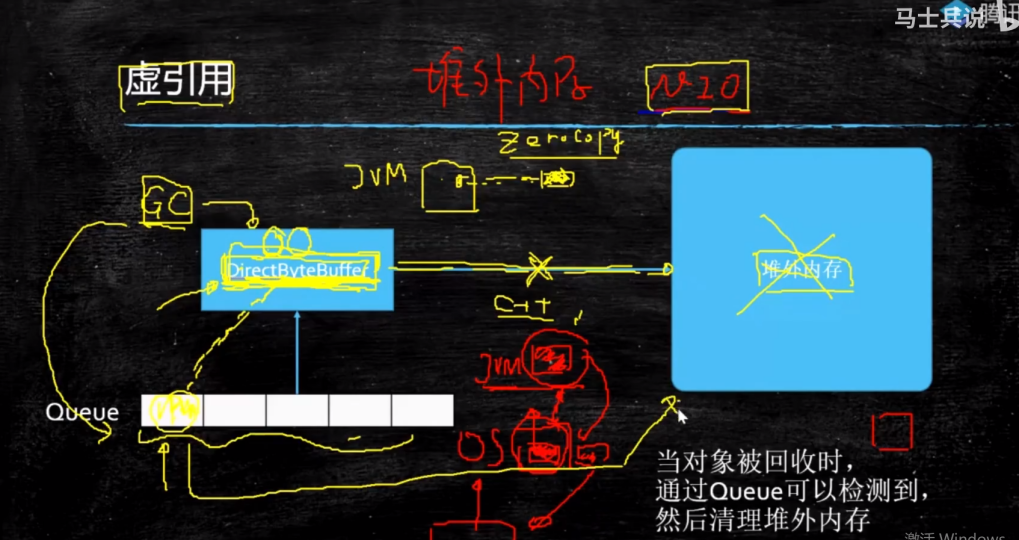


1. 虚引用

其内部提供2个构造函数，一个带queue，一个不带queue。其中queue的意义在于，我们可以在外部对这个queue进行监控。即如果有对象即将被回收，那么相应的reference对象就会被放到这个queue里。我们拿到reference，就可以再作一些事务。







定义：不直接指向堆中对象，而是通过PhantomReference指向堆中对象的引用

GC时间：只要放入Queue中，就会被回收

场景：用于管理堆外内存。某些java对象关联着堆外内存，当java对象回收时，需要把堆外内存也回收。

当堆内对象不用时，将其引用加入Queue中，Queue就检测到，会回收堆外内存（将指向堆外内存的DireByteBuffer加入Queue，此时堆外内存就会被C++回收，不是gc线程回收）。

堆中对象引用将被回收，加入Queue->Queue检测到->回收其指向的堆外内存

保证堆中对象被回收时，其指向的堆外内存也会被回收，防止内存泄漏。