基于Java2平台的引用类使用指南(图) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/273/2021_2022__E5_9F_BA_E4_BA_8EJava_c104_273476.htm Java 2 平台引入了 java.lang.ref 包，其中包括的类可以让您引用对象，而不将它们留在内存中。这些类还提供了与垃圾收集器（garbage collector）之间有限的交互。 Peter Haggar 在本文中分析了 SoftReference、WeakReference 和 PhantomReference 类的功能和行为，并就这些类的使用给出了一些编程风格上的建议。 当在 Java 2 平台中首次引入 java.lang.ref 包（其中包含 SoftReference、WeakReference 和 PhantomReference 类）时，它的实用性显然被过分夸大了。它包含的类可能是有用的，但这些类具有的某些局限性会使它们显得不是很有吸引力，而且其应用程序也将特别局限于解决一类特定的问题。 垃圾收集概述 引用类的主要功能就是能够引用仍可以被垃圾收集器回收的对象。在引入引用类之前，我们只能使用强引用（strong reference）。举例来说，下面一行代码显示的就是强引用 obj： Object obj = new Object(). obj 这个引用将引用堆中存储的一个对象。只要 obj 引用还存在，垃圾收集器就永远不会释放用来容纳该对象的存储空间。 当 obj 超出范围或被显式地指定为 null 时，垃圾收集器就认为没有对这个对象的其它引用，也就可以收集它了。然而您还需要注意一个重要的细节：仅凭对象可以被收集并不意味着垃圾收集器的一次指定运行就能够回收它。由于各种垃圾收集算法有所不同，某些算法会更频繁地分析生存期较短的对象，而不是较老、生存期较长的对象。因此，一个可供收集的对象可能永远也不会被回收。如果

程序在垃圾收集器释放对象之前结束，这种情况就可能会出现。因此，概括地说，您永远无法保证可供收集的对象总是会被垃圾收集器收集。 这些信息对于您分析引用类是很重要的。由于垃圾收集有着特定的性质，所以引用类实际上可能没有您原来想像的那么有用，尽管如此，它们对于特定问题来说还是很有用的类。软引用（soft reference）、弱引用（weak reference）和虚引用（phantom reference）对象提供了三种不同的方式来在不妨碍收集的情况下引用堆对象。每种引用对象都有不同的行为，而且它们与垃圾收集器之间的交互也有所不同。此外，这几个新的引用类都表现出比典型的强引用“ 更弱 ”的引用形式。而且，内存中的一个对象可以被多个引用（可以是强引用、软引用、弱引用或虚引用）引用。在进一步往下讨论之前，让我们来看看一些术语： 强可及对象（strongly reachable）：可以通过强引用访问的对象。 软可及对象（softly reachable）：不是强可及对象，并且能够通过软引用访问的对象。 弱可及对象（weakly reachable）：不是强可及对象也不是软可及对象，并且能够通过弱引用访问的对象。 虚可及对象（phantomly reachable）：不是强可及对象、软可及对象，也不是弱可及对象，已经结束的，可以通过虚引用访问的对象。 清除：将引用对象的 referent 域设置为 null，并将引用类在堆中引用的对象声明为可结束的。 SoftReference 类 SoftReference 类的一个典型用途就是用于内存敏感的高速缓存。SoftReference 的原理是：在保持对对象的引用时保证在 JVM 报告内存不足情况之前将清除所有的软引用。关键之处在于，垃圾收集器在运行时可能会（也可能不会）释放软可及对象。对象是否被释放取决于垃圾收集器的算

法以及垃圾收集器运行时可用的内存数量。 WeakReference 类 WeakReference 类的一个典型用途就是规范化映射（canonicalized mapping）。另外，对于那些生存期相对较长而且重新创建的开销也不高的对象来说，弱引用也比较有用。关键之处在于，垃圾收集器运行时如果碰到了弱可及对象，将释放 WeakReference 引用的对象。然而，请注意，垃圾收集器可能要运行多次才能找到并释放弱可及对象。 PhantomReference 类 PhantomReference 类只能用于跟踪对被引用对象即将进行的收集。同样，它还能用于执行 pre-mortem 清除操作。PhantomReference 必须与 ReferenceQueue 类一起使用。需要 ReferenceQueue 是因为它能够充当通知机制。当垃圾收集器确定了某个对象是虚可及对象时，PhantomReference 对象就被放在它的 ReferenceQueue 上。将 PhantomReference 对象放在 ReferenceQueue 上也就是一个通知，表明 PhantomReference 对象引用的对象已经结束，可供收集了。这使您能够刚好在对象占用的内存被回收之前采取行动。 垃圾收集器和引用交互 垃圾收集器每次运行时都可以随意地释放不再是强可及的对象占用的内存。如果垃圾收集器发现了软可及对象，就会出现下列情况： SoftReference 对象的 referent 域被设置为 null，从而使该对象不再引用 heap 对象。 SoftReference 引用过的 heap 对象被声明为 finalizable。 当 heap 对象的 finalize() 方法被运行而且该对象占用的内存被释放，SoftReference 对象就被添加到它的 ReferenceQueue（如果后者存在的话）。 如果垃圾收集器发现了弱可及对象，就会出现下列情况： WeakReference 对象的 referent 域被设置为 null，从而使该对象不再引用 heap 对象。 WeakReference 引用过

的 heap 对象被声明为 finalizable。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com