Java提供了四种类型的引用：强引用、软引用、弱引用和虚引用。Java中提供这四种引用类型主要目的有两个：第一是可以让程序员通过代码的方式决定某些对象的生命周期；第二是有利于JVM进行垃圾回收。

1. 强引用（StrongReference，任何时候都不会回收）：强引用是使用最普遍的引用。如果一个对象具有强引用，那垃圾回收器绝不会回收它。当内存空间不足时，JVM宁愿抛出OutOfMemoryError错误，使程序终止，也不会随意回收具有强引用的对象来解决内存不足的问题。

Object obj = new Obeject（）；

没有get（）方法。

1. 软引用（SoftReference，内存不足时回收）：一个对象具有软引用，内存空间足够，垃圾回收器就不会回收它；如果内存空间不足，就会回收这些对象的内存。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。软引用可以和一个引用队列（ReferenceQueue，内存不足时回收）联合使用，如果软引用所引用的对象被垃圾回收器回收，JVM就会把这个软引用加入到与之关联的引用队列。

SoftReference<String> str = new SoftReference<> (new String (“hello”));

get（）方法返回引用对象，被回收返回null。

1. 弱引用（WeakReference，执行垃圾回收时回收）：垃圾回收器线程扫描它所管辖的内存区域的过程中，一旦发现具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器时一个优先级很低的线程，因此不一定会很快发现那些具有弱引用的对象。弱引用和软引用一样也可以和引用队列联合使用。

WeakReference<String> str = new WeakReference<> (new String(“hello”));

get（）方法返回引用对象，被回收返回null。

1. 虚引用（PhantomReference，随时可能被回收）：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。虚引用必须和引用队列联合使用。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，就会在回收对象的内存之前，把这个虚引用加入到与之关联的引用队列中。

ReferenceQueue<String> queue = new ReferenceQueue<>();

PhantomReference<String> str = new PhantomReference<String> (new String(“helloe”);

get（）方法返回null。

1. ArrayList<SoftReference> softReferences = new ArrayList<>();

try {

while (true) {

softReferences.add(new SoftReference<>(new byte[1024], referenceQueue));

}

} catch (OutOfMemoryError e) {

e.printStackTrace();

}

上面这段代码，强引用和虚引用会抛出内存溢出错误，软引用和弱引用不会抛出内存溢出错误。

1. 引用队列（ReferenceQueue）：用于保存Reference对象本身，如果Reference引用指向的对象被GC回收，其实Reference已经无效，这种Reference将被放入引用队列，可以在这里进行清除，避免占有空间。
2. 弱引用map（WeakHashMap/WeakHashTable）：弱引用map就是key键是一个弱引用的键，如果key键被回收后，则在get该map中值后，会自动remove掉value。如果key键始终被强引用，则是无法被回收的。这个适合于受key生命周期控制的缓存。value是被强引用的，所以不要让value间接的引用key键，这将导致key始终被强引用。
3. 如何应用软引用避免OOM：假如有一个应用需要读取大量的本地图片，如果每次读取图片都从硬盘读取，则会严重影响性能，但是如果全部加载到内存当中，又可能造成内存溢出，此时使用软引用可以解决问题