****25、Java 中会存在内存泄漏吗，请简单描述。**   
答：理论上Java因为有垃圾回收机制（GC）不会存在内存泄露问题（这也是Java被广泛使用于服务器端编程的一个重要原因）；然而在实际开发中，可能会存在无用但可达的对象，这些对象不能被GC回收，因此也会导致内存泄露的发生。例如Hibernate的Session（一级缓存）中的对象属于持久态，垃圾回收器是不会回收这些对象的，然而这些对象中可能存在无用的垃圾对象，如果不及时关闭（close）或清空（flush）一级缓存就可能导致内存泄露。下面例子中的代码也会导致内存泄露。**

**import java.util.Arrays;**

**import java.util.EmptyStackException;**

**public class MyStack<T> {**

**private T[] elements;**

**private int size = 0;**

**private static final int INIT\_CAPACITY = 16;**

**public MyStack() {**

**elements = (T[]) new Object[INIT\_CAPACITY];**

**}**

**public void push(T elem) {**

**ensureCapacity();**

**elements[size++] = elem;**

**}**

**public T pop() {**

**if(size == 0)**

**throw new EmptyStackException();**

**return elements[--size];**

**}**

**private void ensureCapacity() {**

**if(elements.length == size) {**

**elements = Arrays.copyOf(elements, 2 \* size + 1);**

**}**

**}**

**}**

**上面的代码实现了一个栈（先进后出（FILO））结构，乍看之下似乎没有什么明显的问题，它甚至可以通过你编写的各种单元测试。然而其中的pop方法却存在内存泄露的问题，当我们用pop方法弹出栈中的对象时，该对象不会被当作垃圾回收，即使使用栈的程序不再引用这些对象，因为栈内部维护着对这些对象的过期引用（obsolete reference）。在支持垃圾回收的语言中，内存泄露是很隐蔽的，这种内存泄露其实就是无意识的对象保持。如果一个对象引用被无意识的保留起来了，那么垃圾回收器不会处理这个对象，也不会处理该对象引用的其他对象，即使这样的对象只有少数几个，也可能会导致很多的对象被排除在垃圾回收之外，从而对性能造成重大影响，极端情况下会引发Disk Paging（物理内存与硬盘的虚拟内存交换数据），甚至造成OutOfMemoryError。**