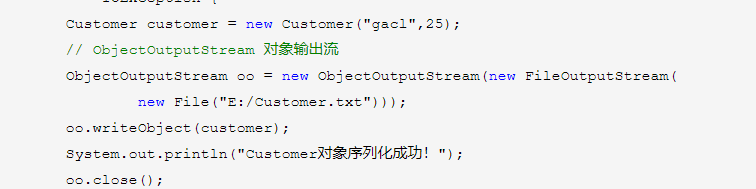
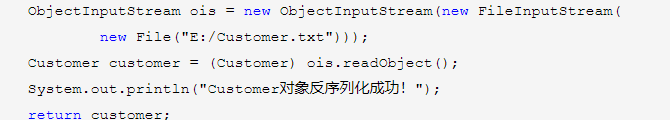
**Serializable接口**

**该接口是一个标识性接口，标注该类的对象是可被序列化的**

* **使用一个输出流来构造一个对象输出流并通过writeObject(Object)方法就可以将对象写出（即保存其状态）；**
* **如果需要反序列化则可以用一个输入流建立对象输入流，然后通过readObject方法从流中读取对象;**





***serialVersionUID***

***serialVersionUID是根据类名、接口名、成员方法及属性等来生成一个64位的哈希字段***

* **序列化操作时会把系统当前类的serialVersionUID写入到序列化文件中，当反序列化时系统会自动检测文件中的serialVersionUID，判断它是否与当前类中的serialVersionUID一致。如果一致说明序列化文件的版本与当前类的版本是一样的，可以反序列化成功，否则就失败**
* **如果没有设置这个值，你在序列化一个对象之后，改动了该类的字段或者方法名之类的，那如果你再反序列化想取出之前的那个对象时就可能会抛出异常，因为你改动了类中间的信息,当修改后的类去反序列化的时候发现该类的serialVersionUID值和之前保存在文件中的serialVersionUID值不一致，所以就会抛出异常**

**而显示的设置serialVersionUID值就可以保证版本的兼容性，如果你在类中写上了这个值，就算类变动了，它反序列化的时候也能和文件中的原值匹配上。而新增的属性则会设置成null，删除的属性则不会显示**

当一个父类实现序列化，子类自动实现序列化，不需要显式实现Serializable接口

**Cloneable接口**

**该接口是一个标识性接口，表示这个类支持被复制**

* **Object类提供一个clone()方法，用于复制对象**
* **没实现该接口，不能调用clone()方法**

**Deque**

* **Deque则是双向队列,它支持在两端插入和移除元素**
* **既可以支持LIFO形式也可以支持LIFO形式，可以把它当成队列或者栈使用**
* **此接口不支持对元素的索引访问**



**Iterator接口**