t

序列化 Serializable

## 1、什么是序列化和反序列化

### Serialization（序列化）是一种将对象以一连串的字节描述的过程；反序列化deserialization是一种将这些字节重建成一个对象的过程。**对象的序列化是基于字节的，不能使用Reader和Writer等基于字符的层次结构**

## 2、什么情况下需要序列化

### a）当你想把的内存中的对象保存到一个文件中或者数据库中时候；

### b）当你想用套接字在网络上传送对象的时候；

**c）当你想通过RMI传输对象的时候；**

#### 实现[serializable接口](https://www.baidu.com/s?wd=serializable%E6%8E%A5%E5%8F%A3&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yzn1cvPA7-PWf1nHDkmWPb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmknjfkn1cd)的作用是就是可以把对象存到字节流，然后可以恢复。所以你想如果你的对象没实现序列化怎么才能进行[网络传输](https://www.baidu.com/s?wd=%E7%BD%91%E7%BB%9C%E4%BC%A0%E8%BE%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yzn1cvPA7-PWf1nHDkmWPb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmknjfkn1cd)呢，要[网络传输](https://www.baidu.com/s?wd=%E7%BD%91%E7%BB%9C%E4%BC%A0%E8%BE%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Yzn1cvPA7-PWf1nHDkmWPb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmknjfkn1cd)就得转为字节流，所以在分布式应用中，你就得实现序列化，比如redis session 负载均衡存储对象，如果你不需要分布式应用，那就没那个必要实现序列化。

简单说就是为了保存在内存中的各种对象的状态（也就是实例变量，不是方法），并且可以把保存的对象状态再读出来。虽然你可以用你自己的各种各样的方法来保存object states，但是[**Java**](javascript:;)给你提供一种应该比你自己好的保存对象状态的机制，那就是序列化

## 3、如何实现序列化

### 将需要序列化的类实现Serializable接口就可以了，Serializable接口中没有任何方法，可以理解为一个标记，即表明这个类可以序列化。

## 4、序列化ID

**序列化 ID 在 Eclipse 下提供了两种生成策略，一个是固定的 1L，一个是随机生成一个不重复的 long 类型数据（实际上是使用 JDK 工具生成），在这里有一个建议，如果没有特殊需求，就是用默认的 1L 就可以，这样可以确保代码一致时反序列化成功。这也可能是造成序列化和反序列化失败的原因，因为不同的序列化id之间不能进行序列化和反序列化。如果使用的是long类型的数据，表明实现序列化类的不同版本间的兼容性。如果你修改了此类, 要修改此值。**

## 5.序列化前和序列化后的对象的关系

### 是 "=="还是equal？ or  是浅复制还是深复制？

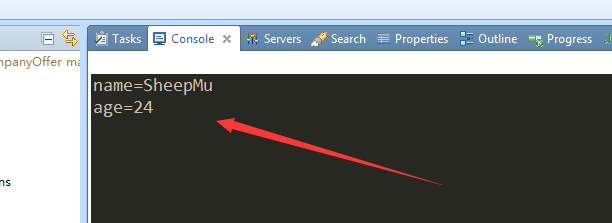
### 答案：深复制，反序列化还原后的对象地址与原来的的地址不同

### 序列化前后对象的地址不同了，但是内容是一样的，而且对象中包含的引用也相同。

## 测试：

|  |
| --- |
| **public** **class** SerializableOne **implements** Serializable  {  **private** **static** **final** **long** serialVersionUID = 1L;  **private** String name="SheepMu";  **private** **int** age=24;    **public** **static** **void** **main**(String[] args) {  **try**  { //以下代码实现序列化  ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream("my.out"));//输出流保存的文件名为 my.out ；ObjectOutputStream能把Object输出成Byte流  SerializableOne myTest=**new** SerializableOne();  oos.writeObject(myTest);  oos.flush(); //缓冲流  oos.close(); //关闭流  } **catch** (FileNotFoundException e)  {  e.printStackTrace();  } **catch** (IOException e)  {  e.printStackTrace();  }  *fan*();//调用下面的 反序列化 代码  }  **public** **static** **void** **fan**()//反序列的过程  {  ObjectInputStream oin = **null**;//局部变量必须要初始化  **try**  {  oin = **new** ObjectInputStream(**new** FileInputStream("my.out"));  } **catch** (FileNotFoundException e1)  {  e1.printStackTrace();  } **catch** (IOException e1)  {  e1.printStackTrace();  }  SerializableOne mts = **null**;  **try** {  mts = (SerializableOne ) oin.readObject();//由Object对象向下转型为MyTest对象  } **catch** (ClassNotFoundException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }  System.out.println("name="+mts.name);  System.out.println("age="+mts.age);  }  } |

## 控制台：



6、序列化会忽略静态变量，即序列化不保存静态变量的状态对象的序列化是操作的堆内存中的数据，静态的变量又称作类变量。类一加载，就初始化了。静态变量在方法区里

## 。static,transient后的变量不能被序列化；

#### 测试

|  |
| --- |
| **public** **class** SerializableStatic **implements** Serializable  {  **private** **static** **final** **long** serialVersionUID = 1L;  **private** String name="SheepMu";  **private** **static** **int** *age*=24;  **public** **static** **void** **main**(String[] args)  {//以下代码实现序列化  **try**  {  ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream("my.out"));//输出流保存的文件名为 my.out ；ObjectOutputStream能把Object输出成Byte流  SerializableStatic myTest=**new** SerializableStatic();  oos.writeObject(myTest);  oos.flush(); //缓冲流  oos.close(); //关闭流  } **catch** (FileNotFoundException e)  {  e.printStackTrace();  } **catch** (IOException e)  {  e.printStackTrace();  }  *fan*();//调用下面的 反序列化 代码  }  **public** **static** **void** **fan**()  {  **new** SerializableStatic().name="SheepMu\_1"; //!!!!!!!!!!!!!!!!重点看这两行 更改部分  *age*=1; //!!!!!!!!!!!!!!!!!!!重点看这两行 更改部分  ObjectInputStream oin = **null**;//局部变量必须要初始化  **try**  {  oin = **new** ObjectInputStream(**new** FileInputStream("my.out"));  } **catch** (FileNotFoundException e1)  {  e1.printStackTrace();  } **catch** (IOException e1)  {  e1.printStackTrace();  }  SerializableStatic mts = **null**;  **try** {  mts = (SerializableStatic ) oin.readObject();//由Object对象向下转型为MyTest对象  } **catch** (ClassNotFoundException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }  System.out.println("name="+mts.name);  System.out.println("age="+mts.*age*);  }  } |

## 控制台：

