Java IO系列课程04

------ 序列化

本文档包含以下内容：

1. 学习目标
2. 什么是序列化
3. 序列化的作用
4. 如何序列化
5. 序列化需要注意
6. 无需序列化的场景
7. 练习题

**前言：**

对象序列化

什么是序列化

序列化的作用（序列化的使用场景）

1、把对象的字节序列化永久的存放在磁盘上，通常放在一个文件夹下

2、在网络上传输对象的字节序列

对象如何序列化

对象的序列化：

目的：将一个具体的对象进行持久化，写入到硬盘上。（注意：静态数据不能被序列化，因为静态数据不在堆内存中，而是在静态方法区中）

Serializable：

用于启动对象的序列化功能，可以强制让指定类具备序列化功能，该接口中没有成员，这是一个标记接口。这个标记接口用于给序列化类提供UID。这个uid是依据类中的成员的数字签名进行运行获取的。如果不需要自动获取一个uid，可以在类中，手动指定一个名称为serialVersionUID id号。依据编译器的不同，或者对信息的高度敏感性。最好每一个序列化的类都进行手动显示的UID的指定。

serialVersionUID（串行化版本统一标识符） 作用：（自行研究）

具体的序列化过程是这样的：序列化操作的时候系统会把当前类的serialVersionUID写入到序列化文件中，当反序列化时系统会去检测文件中的serialVersionUID，判断它是否与当前类的serialVersionUID一致，如果一致就说明序列化类的版本与当前类版本是一样的，可以反序列化成功，否则失败。

serialVersionUID 用来表明类的不同版本间的兼容性。

有两种生成方式： 一个是默认的1L；

另一种是根据类名、接口名、成员方法及属性等来生成一个64位的哈希字段 。

序列化需要注意：

1、在Java中，只要一个类实现了java.io.Serializable接口，那么它就可以被序列化。

2、通过ObjectOutputStream和ObjectInputStream对对象进行序列化及反序列化（非节点流需要通过其他流来链接数据源）

3、虚拟机是否允许反序列化，不仅取决于类路径和功能代码是否一致，

4、序列化并不保存静态变量。

5、要想将父类对象也序列化，就需要让父类也实现Serializable 接口。

6、Transient 关键字的作用是控制变量的序列化，在变量声明前加上该关键字，

可以阻止该变量被序列化到文件中，在被反序列化后，transient 变量的值被设为初始值，

如 int 型的是 0，对象型的是 null。

7、服务器端给客户端发送序列化对象数据，对象中有一些数据是敏感的，比如密码字符串等，

希望对该密码字段在序列化时，进行加密，而客户端如果拥有解密的密钥，只有在客户端进行反序列化时，

才可以对密码进行读取，这样可以一定程度保证序列化对象的数据安全

非序列化使用场景

如何将非静态的数据不进行序列化？用 关键字修饰此变量即可。使用场景：为了安全起见，有时候我们不需要在网络间

传输一些数据（如身份证号码，密码，银行卡号等）

Serializable是一个空的接口，它是怎么保证只有实现了该接口的方法才能进行序列化与反序列化的呢？

其实这个问题也很好回答，我们再回到刚刚ObjectOutputStream的writeObject的调用栈：

writeObject ---> writeObject0--->writeOrdinaryObject--->writeSerialData--->invokeWriteObject

在进行序列化操作时，会判断要被序列化的类是否是Enum、Array和Serializable类型，

如果不是则直接抛出NotSerializableException。

1、如果一个类想被序列化，需要实现Serializable接口。否则将抛出NotSerializableException异常，这是因为，在序列化操作过程中会对类型进行检查，要求被序列化的类必须属于Enum、Array和Serializable类型其中的任何一种。

2、在变量声明前加上该关键字，可以阻止该变量被序列化到文件中。

3、在类中增加writeObject 和 readObject 方法可以实现自定义序列化策略

为什么需要重写

生成新对象的方法