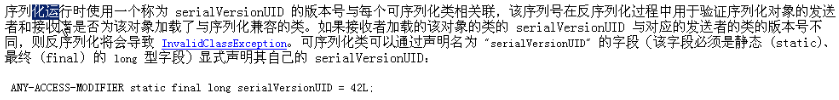
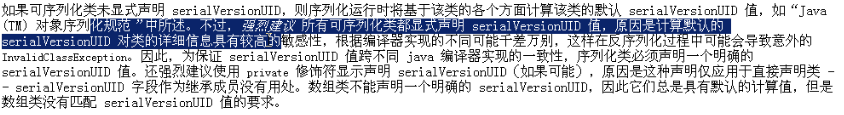
8.操作引用数据类型的IObjectOutputStream（序列化）和ObjectInputStream（反序列化）

1. **引用数据类型指的就是对象。操作对象的IO流**：**ObjectInputSteam（输入流）**与**ObjectOutputStream（输出流）。**
2. **在写对象数据时，需要进行序列化，可以序列化的对象才可以被该流操作，否则，不能被操作，被抛出异常**java.io.NotSerializableException 。
3. **在读取对象数据时，需要进行反序列化。**
4. **接口Serializable**：存在于**java.io包**中，创建的**类**通过实现 java.io.Serializable **接口**以启用其[序列化](http://baike.haosou.com/doc/1053052.html)功能。未实现此接口的类将无法使其任何状态**序列化或反序列化**。可序列化类的所有子类型本身都是**可序列化**的。**序列化接口没有方法或字段，仅用于标识可序列化的语义。**
5. 对象在反序列化时，必须具备该对象类的**class文件**，且对象和class类必须版本相同。否则会抛出**InvalidClassException异常**。



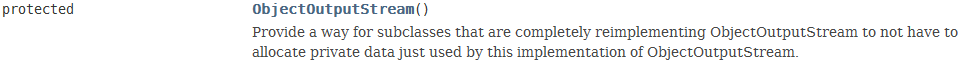
**接口Serializable存在的意义**：用于给被序列化的类加入**ID号**，用于判断类和对象是否是同一个版本。如果不是同一个版本，则会抛出java.io.InvalidClassException异常。



编译器可以根据该类的内容，算出ID号，但是由于不同的编译器，可能会导致不同的ID号,所以建议显式地声明ID号：

private static final long serialVersionUID = 1343L;

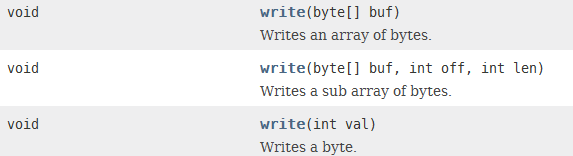
1. **ObjectOutputStream对象输出流**：
2. 构造方法：2个
3. 空参数：注意是protected修饰的。



1. **输出流作为参数**：



1. 成员方法：
2. write方法：



1. **writeXxx方法**：**Int 、Double、Float、Long、Boolean、Byte、Char、short，还有writeChars、writeBytes**





1. writeFields方法：



1. **writeObject方法**：要想操作对象，被操作的对象的类必须实现**Serializable序列化接口**，这个接口没有要实现的方法，只是个标记性接口。



1. writeUnshared（Object obj）



1. **writeUTF（String str）**



1. **close方法、flush方法、reset方法等**
2. **读写对象的意义**：使对象的生命周期延长，存到硬盘中，需要时直接从硬盘中直接读取出来。**一般存储对象的文件的扩展名为 .object 。**
3. **ObjectOutputStream**可以将Java对象的基本数据类型和图形写入OutputStream，可以使用ObjectInputStream读取对象，通过在流中使用文件可以**实现对象的永久存储**。如果流时网络套接字，则可以在另一台主机上或另一个进程中**重构对象**。
4. ObjectInputStream对之前使用ObjectOutputStream写入的基本数据和对象进行**反序列化**。
5. ObjectInputStream流：

总体上来说是和ObjectOutputStream流是对应的。

1. 构造方法：2个
2. 无参数：



1. InputStream流做参数：**实现对输入流的包装，对其添加新功能。**



1. 成员方法：主要是read方法。
2. available方法：



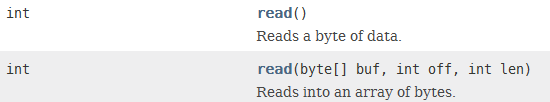
1. close



1. skipBytes



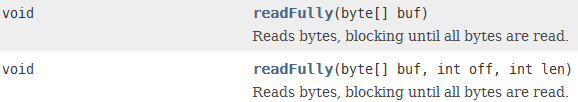
1. read方法：
   1. 一般的read方法：



* 1. readXxxx ： 8个基本数据类型
  2. readObject：



* 1. readFully



* 1. readUTF



* 1. readFields



1. **对象中的静态成员变量**，不是对象的特有内容，不会被写入。当然读取的时候也不是上次的值了，而是默认值。
2. 注意事项：
   1. **对象必须首先序列化，然后反序列化；**
   2. **反序列化的对象的顺序必须与序列化的顺序一致，否则错误。**
   3. 不是所有的对象都可以序列化，只有是实现了java.io.Serializable接口才可以。
   4. 不是所有的属性都需要序列化，不想序列化的属性利用关键字**transient 修饰即可。**此成员变量在反序列化时，读取的是**此变量的默认值**。
3. **transient** ['trænzɪənt] n. 瞬变现象；过往旅客；候鸟adj. 短暂的；路过的
4. **关键字：transient：如果对象中的某个成员变量不想实现序列化即不想持久化（不想存储到硬盘中），可**利用**transient关键字**修饰。则此成员变量在反序列化时，读取的是**此变量的默认值**。

public **transient** String name;

private **transient** int age;

读取的值为 null 和 0.

1. **序列化和反序列化的示例**：
   1. 操作一个普通对象：

序列化：

**private** **static** **void** writeObjec(File file) **throws** IOException,

FileNotFoundException {

ObjectOutputStream oos = **new** ObjectOutputStream(**new** BufferedOutputStream(**new** FileOutputStream(file)));

Employee employee = **new** Employee("朱耀金","男",55555.32f);

Boss boss = **new** Boss("朱吃金","男",66666.32f);

oos.writeObject(employee);

oos.writeObject(boss);

oos.close();

}

反序列化：

**private** **static** **void** readObjec(File file) **throws** IOException,

FileNotFoundException, ClassNotFoundException {

ObjectInputStream ois = **new** ObjectInputStream(**new** BufferedInputStream(**new** FileInputStream(file)));

**System.*out*.println((Employee) ois.readObject());**

**System.*out*.println((Boss)ois.readObject());//两者顺序不可换，否则出现类转换异常**java.lang.ClassCastException

ois.close();

}

测试：

File file = **new** File("e:\\aaa.txt");

*writeObjec*(file);

*readObjec*(file);

结果：

* 1. 操作一个字节数组（数组也是一个对象，也可以利用Object接收）

序列化：

**private** **static** **void** writeObjec(File file) **throws** IOException,

FileNotFoundException {

ObjectOutputStream oos = **new** ObjectOutputStream(**new** BufferedOutputStream(**new** FileOutputStream(file)));

**byte**[] bs = **new** **byte**[] {33,44,22,3,21,42};

oos.writeObject(bs);

oos.close();

}

反序列化：

**private** **static** **void** readObjec(File file) **throws** IOException,

FileNotFoundException, ClassNotFoundException {

ObjectInputStream ois = **new** ObjectInputStream(**new** BufferedInputStream(**new** FileInputStream(file)));

Object obj = (**byte**[]) ois.readObject();

System.***out***.println(Arrays.*toString*((**byte**[])obj));

ois.close();

}

测试：

**public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException {

File file = **new** File("e:\\aaa.txt");

*writeObjec*(file);

*readObjec*(file);

}