操作基本数据类型的DataInputStream和DataOutputStream

1. **DataInputStream和DataOutputStream**是一对IO流，可以操作**基本数据类型和String数据**。也**是一种处理流，二进制数据+类型 ==》数据IO流。**
2. **DataOutputStream流**介绍：
3. **构造方法**:只有一个，对输出流添加**了新的功能**。



1. 成员方法:
2. **size方法**

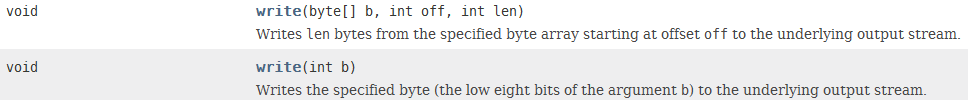


1. **flush方法**：



1. **write方法**:

①传统的write方法：



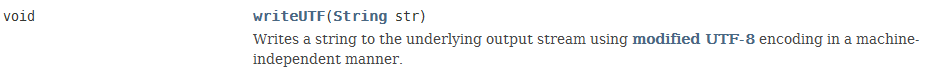
②写基本数据类型：writeXxx方法

还有**writeBytes和writeChars**





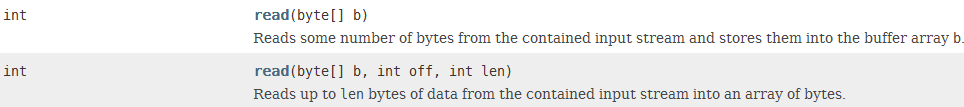
③writeUTF方法:



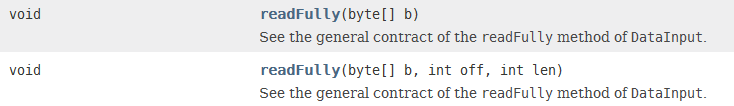
1. **DataInputStream流**介绍：
2. **构造方法**：1个,对读取流进行添加新的功能。



1. 成员方法：
2. 传统的read方法



1. **readXxx方法**：读取基本数据类型。
2. readFully方法：

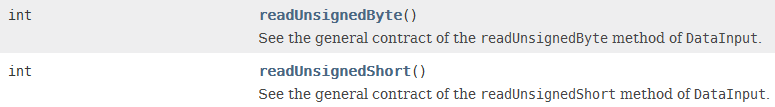


1. **readUTF方法**：





1. readUnsignedByte



1. skipBytes



1. 对于**writeUTF和readUTF方法**：写到文件中时，在开头处有特殊标记字符，因此用writeUTF写的，只能利用readUTF读取。
2. **write方法和writeInt方法的区别**：write方法会自动舍去高3个字节，而只保留最低8位，但是**writeInt方法**可以把4个字节都写到目的地。
3. **练习：**
   1. **写入到文件，从文件中读取数据。**

**写数据：**

**private** **static** **void** writeData(File file) **throws** FileNotFoundException, IOException {

DataOutputStream dos = **new** DataOutputStream(**new** BufferedOutputStream(**new** FileOutputStream(file)));

dos.writeDouble(11111.333);

dos.writeLong(3333L);

dos.writeUTF("面试也是一种学习");

dos.close();

}

读数据：

**private** **static** **void** readData(File file) **throws** FileNotFoundException,

IOException {

DataInputStream dis = **new** DataInputStream(**new** BufferedInputStream(**new** FileInputStream(file)));

**double** d = dis.readDouble();

**long** l = dis.readLong();

String string = dis.readUTF();

System.***out***.println(d+" "+l+" "+string);

}

测试：

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

*writeData*(file);

*readData*(file);

* 1. **写入到数组中，然后从数组中读取。**

**需要利用数组流：字节数组流或字符数组流。**

**写到数组中**：**并返回数据。**

**private** **static** **byte**[] writeDataToArray() **throws** IOException {

ByteArrayOutputStream bos = **new** ByteArrayOutputStream();

DataOutputStream dos = **new** DataOutputStream(**new** BufferedOutputStream(

bos));

**byte**[] result = **null**;

dos.writeByte(444);

dos.writeDouble(444.9);

dos.writeUTF("愈挫愈勇");

**dos.flush();// 必须要刷新，虽然说close方法会自动刷新但是，**

**// 在这里需要获取数组，不能close且必须刷新。**

**result = bos.toByteArray();//只能利用bos，因为dos没有这个方法**

dos.close();

**return** result;

}

**从数组中读**：

**private** **static** **void** readDataArray(**byte**[] bs) **throws** IOException {

DataInputStream dis = **new** DataInputStream(**new** BufferedInputStream(

**new** ByteArrayInputStream(bs)));

**byte** b = dis.readByte();

**double** d = dis.readDouble();

String str = dis.readUTF();

System.***out***.println(b + "-->" + d + "-->" + str);

dis.close();

}

**测试**：

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

**byte**[] result = *writeDataToArray*();

System.***out***.println(Arrays.*toString*(result));

*readDataArray*(result);

}

1. 使用**数据IO流**的时候需要注意：
2. 读取的文件必须存在，且文件内容中含有指定类型的的数据，否则抛出异常java.io.EOFException ，此异常就是超出边界异常，找不到指定数据。
3. 读取的数据类型的顺序要与写入的数据类型数据一致，否则出现读错；

如：**double** d = dis.readDouble();

**long** l = dis.readLong();

String string = dis.readUTF();

与dos.writeDouble(11111.333);

dos.writeLong(3333L);

dos.writeUTF("面试也是一种学习");

必须顺序一致，读与写 的顺序一致。

1. 可以**操作基本数据类型**的**IO流**都实现了**接口DataInput和DataOutput**：
2. DataInputStream和DataOutputStream就分别实现了这两个接口。
3. ObjectInput和ObjectOutput这两个接口分别是这两个接口的子接口，

又**ObjectInputStream和ObjectOutputStream**也分别实现了ObjectInput和ObjectOutput这两个接口，且分别继承于InputStream和OutputStream。因此对象输入输出流也可以直接**操作基本数据类型。**

1. **RandomAccessFile**既可以读也可以写，既实现了DataInput也实现了DataOutput，直接继承于**java.lang.Object类，随机访问文件就相当于一个数组，可以指定指针，分段读写**。