12 IO流_转换流 && 数据流 && 对象流

JAVAEE高级

一：转换流描述

> 1、转换流提供了在字节流和字符流之间的转换
>
> 2、Java API提供了两个转换流InputStreamReader和OutputStreamWriter
>
> 3、字节流中的数据都是字符时，转成字符流操作更高效

```
1.      @Test
2.      public void getVoid() throws IOException {
3.          InputStreamReader isr = new InputStreamReader(new FileInputStre
    am("D:\\IO\\修改后.txt"), "utf-8"); //指定码表读字符
4.          OutputStreamWriter osw = new OutputStreamWriter(new FileOutputS
    tream("D:\\IO\\修改后11啊啊.txt"), "utf-8");  //指定码表写字符
5.          int c;
6.          while((c = isr.read()) != -1) {
7.              osw.write(c);
8.          }
9.          isr.close();
10.         osw.close();
11.     }
```

> 3、使用缓冲流来进行完成读写

```
1.  @Test
2.      public void getVoid() throws IOException {
3.          BufferedReader br = // 更高效的读
4.                  new BufferedReader(new InputStreamReader(new
    FileInputStream("D:\\IO\\修改后.txt"), "utf-8"));
5.          BufferedWriter bw = // 更高效的写
6.                  new BufferedWriter(new OutputStreamWriter(new
    FileOutputStream("D:\\IO\\修改后1.txt"), "utf-8"));
```

```
            int c;
            while ((c = br.read()) != -1) {
                bw.write(c);
            }
            br.close();
            bw.close();
        }
```

二：数据流描述

> 1、为了方便地操作Java语言的基本数据类型的数据，可以使用数据流。
>
> 2、数据流有两个类：(用于读取和写出基本数据类型的数据）
> DataInputStream 和 DataOutputStream

```
    @Test
    public void getVoid() throws IOException {
        //写的文件是看不懂的
        DataOutputStream data = new DataOutputStream(new FileOutputStream("D:\\IO\\修改后1.txt"));
        data.writeUTF("年龄");
        data.writeInt(1212);
        data.close();

        DataInputStream DOS = new DataInputStream(new FileInputStream("D:\\IO\\修改后1.txt"));
        String name = DOS.readUTF();
        int age = DOS.readInt();
        System.out.println(name + ":" + age);
    }
```

三：对象流

> 1、用于存储和读取对象的处理流ObjectInputStream和OjbectOutputSteam
>
> 2、它的强大之处就是可以把Java中的对象写入到数据源中，也能把对象从数据源中还原回来

3、ObjectOutputStream和ObjectInputStream不能序列化static和transient修饰的成员变量

4、序列化
　　用ObjectOutputStream类将一个Java对象写入IO流中

5、反序列化
　　用ObjectInputStream类从IO流中恢复该Java对象

1、对象序列化机制

1、把内存中的Java对象转换成平台无关的二进制流，从而允许把这种二进制流持久地保存在磁盘上

2、或者通过网络将这种二进制流传输到另一个网络节点

3、其它程序获取了这种二进制流，就可以恢复成原来的Java对象(反序列化)

4、序列化的好处在于可将任何实现了Serializable接口的对象转化为字节数据，使其在保存和传输时可被还原

5、如果需要让某个对象支持序列化机制，则必须让其类是可序列化的需要进行实现接口(Serializable、Externalizable)

6、凡是实现Serializable接口的类都有一个表示序列化版本标识符的静态变量`private static final long serialVersionUID`用来表明类的不同版本间的兼容性

注意:
如果某个类的字段不是基本数据类型或 String 类型，而是另一个引用类型，那么这个引用类型必须是可序列化的，否则拥有该类型的 Field 的类也不能序列化

```
1.
2.  public class Person implements Serializable {
3.      /**
4.       * @Fields serialVersionUID : TODO(用一句话描述这个变量表示什么)
5.       */
6.      private static final long serialVersionUID = 1L;
7.      private String name;
8.      private int age;
```

```
    //get set 构造 toString...
    }
```

```
@Test
    public void getVoid() throws IOException, ClassNotFoundException {
        //将对象序列化到本地（磁盘）
        ObjectOutputStream object = new ObjectOutputStream(new FileOutputStream("D:\\IO\\person.txt"));
        Person person = new Person("李四", 12);
        object.writeObject(person);
        object.flush();
        object.close();
        //对象的反序列化
        ObjectInputStream   In = new ObjectInputStream(new FileInputStream("D:\\IO\\person.txt"));
        Person St =(Person)In.readObject();   //需要进行强制转换
          System.out.println(St);
    }
```