# 字符流转向字节流的桥梁

Ascii

中

GBK码表

中 [-42, -48]

UTF-8 码表

中 [-28, -72, -83]

|  |
| --- |
| **public** **class** ConverterDemo {  **public** **static** **void** main(String[] args) {  //创建字符流转换字节流的桥梁的对象  OutputStreamWriter ow = **null**;  **try** {  //ow = new OutputStreamWriter(new FileOutputStream("b.txt"));  //字符流通向字节流的桥梁可以指定存储的编码  ow = **new** OutputStreamWriter(**new** FileOutputStream("b.txt"),"GBK");  ow.write("中");  ow.flush();  } **catch** (Exception e) {  e.printStackTrace();  }**finally** {  **try** {  **if**(ow != **null**){  ow.close();  }  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  } |

# Properties

Properties 类表示了一个持久的属性集。Properties 可保存在流中或从流中加载。属性列表中每个键及其对应值都是一个字符串。

特点：

1.继承于Hashtable，是线程安全的键值对存储结构

2. Properties 可保存在流中或从流中加载

3. 只能保存字符串的键值对。

实战应用

|  |
| --- |
| **public class** PropTest6 {    **public static void** main(String[] args) {   *//创建一个Properties对象* Properties prop = **new** Properties();    InputStream in = **null**;  **try** {  *//in = new FileReader("src/names.properties");* in = PropTest6.**class**.getClassLoader().getResourceAsStream(**"names.properties"**);  *//从字节流中来加载数据到属性对象中* prop.load(in);  System.***out***.println(prop);   } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **if**(in != **null**){  **try** {  in.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  } } |

# 序列化流

序列化流：

对象输出流：

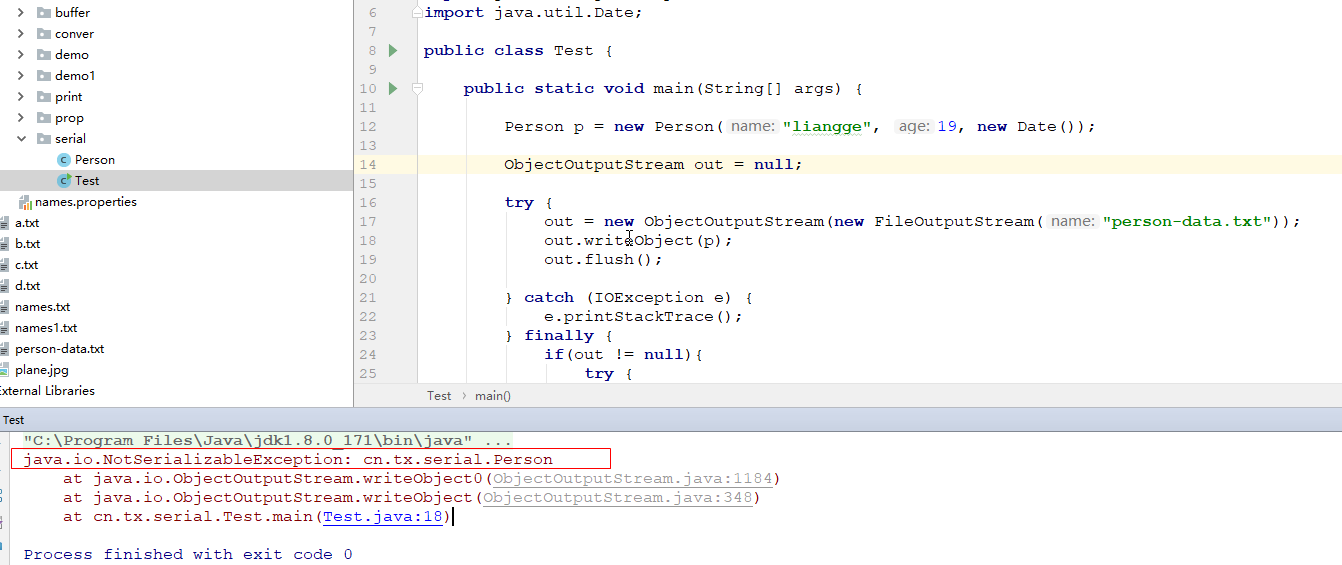
ObjectOutputStream 将 Java 对象写入 OutputStream。

对象输入流：

ObjectInputStream 对以前使用 ObjectOutputStream 写入的对象进行反序列化。

## ObjectOutputStream

我们要序列化的对象需要实现序列化接口



实现



我们一般会提供一个***serialVersionUID***

某个类序列化之后，如果类发生了 ，那么依然可以反序列化。

如果要是对多个对象来做序列化，一定要放在集合中。

# 字符编码

存储:

在计算机中存储字符都是存储的字符所对应的数值以二进制的形式表示。

展示：

去相关的编码表中去查找该值（存储的值）所对应的字符。

Ascii

a 97

b 98

c 99

GBK :用于存储中文的码表

中 [-42, -48]

UTF-8: 最大的一张码表包含了所有的字符标准

中 [-28, -72, -83]

用什么编码，就要用什么解码