第10天 IO流

今日内容介绍

* 标准输入流 & 转换流 & 打印流
* 对象操作流
* Properties集合

# 标准输入流 & 转换流 & 打印流

## 标准输入输出流

public static final InputStream in:标准输入流

public static final PrintStream out:标准输出流

**package** com.itheima\_05;

/\*

\* 标准输入输出流：

public static final InputStream in：字节输入流,用来读取键盘录入的数据

public static final int x;

InputStream is = System.in;

Scanner sc = new Scanner(System.in);

public static final PrintStream out：字节输出流,将数据输出到命令行

System.out.println();

\*/

**public** **class** SystemInOutDemo {

}

## 转换流

### OutputStreamWriter:将字节输出流转换为字符输出流

#### 案例代码一:

**package** com.itheima\_05;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.io.OutputStream;

**import** java.io.OutputStreamWriter;

**import** java.io.Writer;

/\*

\* 需求：读取项目根目录下的SystemInOutDemo.java，并输出到命令行

\*

\* 数据源：项目根目录下的SystemInOutDemo.java BufferedReader

\* 目的地：命令行 System.out

\*

\*

\* 由于标准输出流是一个字节输出流，所以只能输出字节或者字节数组，但是我们读取到的数据则是字符串，如果想进行输出还需要转换成字节数组

\* 我们要想通过标准输出流输出字符串，把标准输出流转换成一种字符输出流即可，OutputStreamWriter

\*

\* OutputStreamWriter(OutputStream out) :转换流，把字节输出流转换成字符输出流

\*

\*

\*/

**public** **class** OutputStreamWriterDemo {

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

//method2();

//创建输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("SystemInOutDemo.java"));

//创建输出流对象

//OutputStream os = System.out;

//Writer w = new OutputStreamWriter(System.out);//多态，父类型引用指向子类对象

//BufferedWriter bw = new BufferedWriter(w);

BufferedWriter bw = **new** BufferedWriter(**new** OutputStreamWriter(System.*out*));

//进行数据的读写

String line;//用于存储读取到的数据

**while**((line = br.readLine()) != **null**) {

bw.write(line);

bw.newLine();

}

//释放资源

bw.close();

br.close();

}

**private** **static** **void** method2() **throws** FileNotFoundException, IOException {

//创建输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("SystemInOutDemo.java"));

//创建输出流对象

//OutputStream os = System.out;

Writer w = **new** OutputStreamWriter(System.*out*);//多态，父类型引用指向子类对象

//进行数据的读写

String line;//用于存储读取到的数据

**while**((line = br.readLine()) != **null**) {

w.write(line);

w.write("\r\n");

}

//释放资源

w.close();

br.close();

}

**private** **static** **void** method() **throws** FileNotFoundException, IOException {

//创建输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("SystemInOutDemo.java"));

//创建输出流对象

OutputStream os = System.*out*;

String line;//用于存储读取到的数据

**while**((line = br.readLine()) != **null**) {

os.write(line.getBytes());

os.write("\r\n".getBytes());

}

//释放资源

os.close();

br.close();

}

}

### InputStreamReader:将字节输入流转换为字符输入流

#### 案例代码二:

**package** com.itheima\_05;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.InputStreamReader;

**import** java.io.Reader;

/\*

\* 需求：读取键盘录入的数据，并输出到项目根目录下的a.txt文件中

\*

\* 数据源：读取键盘录入的数据 System.in

\* 目的地：项目根目录下的a.txt FileWriter

\*

\*

\*

\* 转换流：需要把字节输入流转换成字符输入流，InputStreamReader

\* InputStreamReader(InputStream in)

\*/

**public** **class** InputStreamReaderDemo {

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

//创建输入流对象

InputStream is = System.*in*;

Reader r = **new** InputStreamReader(is);

//创建输出流对象

FileWriter fw = **new** FileWriter("a.txt");

//读写数据

**char**[] chs = **new** **char**[1024];

**int** len;

**while**((len = r.read(chs)) != -1) {

fw.write(chs,0,len);

fw.flush();

}

//释放资源

fw.close();

is.close();

}

**private** **static** **void** method() **throws** IOException {

//创建输入流对象

InputStream is = System.*in*;

//创建输出流对象

FileWriter fw = **new** FileWriter("a.txt");

//读写数据

**byte**[] bys = **new** **byte**[1024];

**int** len;//用于存储读取到的字节个数

**while**((len = is.read(bys)) != -1) {

fw.write(**new** String(bys,0,len));

fw.flush();

}

//释放资源

fw.close();

is.close();

}

}

## 打印流

打印流添加输出数据的功能，使它们能够方便地打印各种数据值表示形式.

字符打印流 PrintWriter

void print(String str): 输出任意类型的数据，

void println(String str): 输出任意类型的数据，自动写入换行操作

### 案例代码三:

**package** com.itheima\_06;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.PrintWriter;

/\*

\* 打印流：

\* PrintStream

\* PrintWriter

\* 可以自动换行，println()

\* 不能输出字节，但是可以输出其他任意类型

\* 通过某些配置，可以实现自动刷新（只有在调用 println、printf 或 format才有用）

\* 也是包装流，不具备写出功能

\* 可以把字节输出流转换成字符输出流

\*

\* 注意：只能输出不能输入

\*

\*

\*/

**public** **class** PrintWriterDemo {

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

//创建打印流对象

PrintWriter pw = **new** PrintWriter("b.txt");

//写出数据

pw.write("hello");

pw.write("world");

pw.write("java");

//释放资源

pw.close();

}

}

### 案例代码四:

利用打印流实现自动换行与自动更新

**package** com.itheima\_06;

**import** java.io.FileNotFoundException;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintWriter;

/\*

\* 打印流的特有功能：

\* 自动换行 使用方法println()实现自动换行

\* 自动刷新 创建PrintWriter对象时启动自动刷新开关，并且使用println等3个方法可以实现自动刷新

\*

\* 注意：创建FileWriter对象时boolean参数是是否追加，

\* 而创建打印流对象的boolean类型参数是是否自动刷新

\*/

**public** **class** PrintWriterDemo2 {

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

//method();

//创建打印流对象

//PrintWriter pw = new PrintWriter("d.txt");

PrintWriter pw = **new** PrintWriter(**new** FileWriter("d.txt"),**true**);

pw.println("hello");

pw.println("world");

pw.println("java");

//释放资源

//pw.close();

}

**private** **static** **void** method() **throws** FileNotFoundException {

//创建打印流对象

PrintWriter pw = **new** PrintWriter("c.txt");

/\*pw.write("hello");

pw.write("world");

pw.write("java");\*/

pw.print("hello");

pw.println("world");

pw.println("java");

//释放资源

pw.close();

}

}

### 案例代码五:

利用打印流将根目录下的SystemInOutDemo.java复制到d:\\SystemInOutDemo.java下

**package** com.itheima\_06;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintWriter;

/\*

\* 使用打印流复制文本文件

\*

\* 数据源 SystemInOutDemo.java BufferedReader

\* 目的地 d:\\SystemInOutDemo.java PrintWriter

\*

\*/

**public** **class** PrintWriterDemo3 {

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

//创建输入流对象

BufferedReader br = **new** BufferedReader(**new** FileReader("SystemInOutDemo.java"));

//创建打印流对象

PrintWriter pw = **new** PrintWriter(**new** FileWriter("d:\\SystemInOutDemo.java"),**true**);

String line;//用于存储读取到的每行数据

**while**((line = br.readLine()) != **null**) {

pw.println(line);

}

//释放资源

pw.close();

br.close();

}

}

# 对象操作流

## 概述

用于从流中读取对象的

ObjectInputStream 称为 反序列化流,利用输入流从文件中读取对象

ObjectOutputStream 称为 序列化流,利用输出流向文件中写入对象

特点：用于操作对象。可以将对象写入到文件中，也可以从文件中读取对象。

**package** com.itheima\_07;

/\*

\* 对象操作流：可以用于读写任意类型的对象

\* ObjectOutputStream

\* writeObject

\* ObjectOutputStream(OutputStream out)

\* ObjectInputStream

\* readObject

\* ObjectInputStream(InputStream in)

\*

\* 注意：

\* 使用对象输出流写出对象，只能使用对象输入流来读取对象

\* 只能将支持 java.io.Serializable 接口的对象写入流中

\*

\*/

**public** **class** ObjectOutputStreamDemo2 {

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

}

}

## 利用序列化流读写对象

**package** com.itheima\_07;

**import** java.io.Serializable;

**public** **class** Student **implements** Serializable {

/\*\*

\*

\*/

String name;

**int** age;

**public** Student(String name,**int** age) {

**this**.name = name;

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Student [name=" + name + ", age=" + age +"]";

}

}

**package** com.itheima\_07;

**import** java.io.EOFException;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

/\*

\* 使用对象输出流和读对象输入流写对象

\* Exception in thread "main" java.io.NotSerializableException: com.itheima\_07.Student

\* Serializable:序列号，是一个标识接口，只起标识作用，没有方法

\* 当一个类的对象需要IO流进行读写的时候，这个类必须实现该接口

\*

\* Exception in thread "main" java.io.EOFException:当输入过程中意外到达文件或流的末尾时，抛出此异常。

\*

\*/

**public** **class** ObjectOutputStreamDemo {

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

//method();

//创建对象输入流的对象

ObjectInputStream ois = **new** ObjectInputStream(**new** FileInputStream("a.txt"));

//读取对象

/\*Object obj = ois.readObject();

System.out.println(obj);

Object obj2 = ois.readObject();

System.out.println(obj2);

Object obj3 = ois.readObject();

System.out.println(obj3);\*/

**try** {

**while**(**true**) {

Object obj = ois.readObject();

System.*out*.println(obj);

}

} **catch**(EOFException e) {

System.*out*.println("读到了文件的末尾");

}

//释放资源

ois.close();

}

**private** **static** **void** method() **throws** IOException, FileNotFoundException {

//创建对象输出流的对象

//FileOutputStream fos = new FileOutputStream("a.txt");

//ObjectOutputStream oos = new ObjectOutputStream(fos);

ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream("a.txt"));

//创建学生对象

Student s = **new** Student("zhangsan",18);

Student s2 = **new** Student("lisi",19);

//写出学生对象

oos.writeObject(s);

oos.writeObject(s2);

//释放资源

oos.close();

}

}

### 案例代码六:

**package** com.itheima\_02;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

/\*

\* 使用字符流复制文本文件

\*

\* 数据源 IODemo.java

\* 目的地 d:\\IODemo.java

\*

\*/

**public** **class** FileCopyDemo {

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

//创建字符输入流对象

FileReader fr = **new** FileReader("IODemo.java");

//创建字符输出流对象

FileWriter fw = **new** FileWriter("d:\\IODemo.java");

//一次读写一个字符

/\*int ch;

while((ch = fr.read()) != -1) {

fw.write(ch);

fw.flush();

}\*/

//一次读写一个字符数组

**int** len;//用于存储读到的字符个数

**char**[] chs = **new** **char**[1024];

**while**((len = fr.read(chs)) != -1) {

fw.write(chs,0,len);

fw.flush();

}

//释放资源

fw.close();

fr.close();

}

}

## 解决对象输入流读取对象出现异常的问题

### 案例代码七:

**package** com.itheima\_07;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.util.ArrayList;

/\*

\* 解决对象输入流读取对象出现异常的问题

\*

\*/

**public** **class** ObjectOutputStreamDemo3 {

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

//method();

//创建对象输入流的对象

ObjectInputStream ois = **new** ObjectInputStream(**new** FileInputStream("b.txt"));

//读取数据

Object obj = ois.readObject();

//System.out.println(obj);

//向下转型，获取具体的子类对象

ArrayList<Student> list = (ArrayList<Student>) obj;

**for** (Student student : list) {

System.*out*.println(student);

}

//释放资源

ois.close();

}

**private** **static** **void** method() **throws** IOException, FileNotFoundException {

//创建对象输出流的对象

ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream("b.txt"));

//创建集合对象

ArrayList<Student> list = **new** ArrayList<Student>();

//添加学生对象

list.add(**new** Student("wangwu",30));

list.add(**new** Student("zhaoliu",28));

//写出集合对象

oos.writeObject(list);

//释放资源

oos.close();

}

}

## 解决读写对象版本不一致问题

### 案例代码八:

**package** com.itheima\_07;

**import** java.io.Serializable;

**public** **class** Student **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** *serialVersionUID* = 6361890890437825953L;

String name;

**int** age;

String gender;

**public** Student(String name,**int** age) {

**this**.name = name;

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Student [name=" + name + ", age=" + age + ", gender=" + gender + "]";

}

}

**package** com.itheima\_07;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

/\*

\* 解决对实现序列化接口出现的黄色警告问题

\* Exception in thread "main" java.io.InvalidClassException

\* 当 Serialization 运行时检测到某个类具有以下问题之一时，抛出此异常。

该类的序列版本号与从流中读取的类描述符的版本号不匹配

该类包含未知数据类型

该类没有可访问的无参数构造方法

\*

\*/

**public** **class** ObjectOutputStreamDemo4 {

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

//method();

*method2*();

}

//读取学生对象

**private** **static** **void** method2() **throws** IOException, FileNotFoundException, ClassNotFoundException {

//创建对象输入流的对象

ObjectInputStream ois = **new** ObjectInputStream(**new** FileInputStream("c.txt"));

//读取对象

Object obj = ois.readObject();

System.*out*.println(obj);

//释放资源

ois.close();

}

//写出学生对象

**private** **static** **void** method() **throws** IOException, FileNotFoundException {

//创建对象输出流的对象

ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream("c.txt"));

//创建的学生对象

Student s = **new** Student("qianqi",28);

//写出学生对象

oos.writeObject(s);

//释放资源

oos.close();

}

}

# Properties集合

## Properties介绍

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。

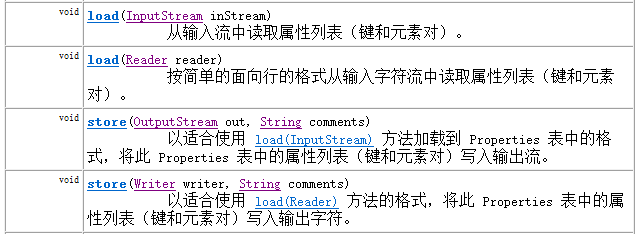
特点：

1、Hashtable的子类，map集合中的方法都可以用。

2、该集合没有泛型。键值都是字符串。

3、它是一个可以持久化的属性集。键值可以存储到集合中，也可以存储到持久化的设备(硬盘、U盘、光盘)上。键值的来源也可以是持久化的设备。

4、有和流技术相结合的方法。



## 利用Properties存储键值对

### 案例代码九:

**package** com.itheima\_08;

**import** java.util.Map;

**import** java.util.Properties;

**import** java.util.Set;

/\*

\* Properties：表示了一个持久的属性集，属性列表中每个键及其对应值都是一个字符串

\*

\* 构造方法：

\* Properties()

\*/

**public** **class** PropertiesDemo2 {

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

//创建属性列表对象

Properties prop = **new** Properties();

//添加映射关系

prop.put("CZBK001", "zhangsan");

prop.put("CZBK002", "lisi");

prop.put("CZBK003", "wangwu");

//遍历属性列表

//获取所有的key，通过key获取value

Set<Object> keys = prop.keySet();

**for** (Object key : keys) {

Object value = prop.get(key);

System.*out*.println(key + "=" + value);

}

System.*out*.println("------------------");

//获取所有的结婚证对象

Set<Map.Entry<Object,Object>> entrys = prop.entrySet();

**for** (Map.Entry<Object, Object> entry : entrys) {

Object key = entry.getKey();

Object value = entry.getValue();

System.*out*.println(key + "=" + value);

}

}

}

## Properties与流结合使用

### 案例代码十:

**package** com.itheima\_08;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Properties;

/\*

\* Properties和IO流结合的功能：

void load(Reader reader)

void list(PrintWriter out)

void store(Writer writer, String comments)

\*

\*/

**public** **class** PropertiesDemo2 {

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

//method();

//method2();

//创建属性列表对象

Properties prop = **new** Properties();

//添加映射关系

prop.setProperty("CZBK001", "zhangsan");

prop.setProperty("CZBK002", "lisi");

prop.setProperty("CZBK003", "wangwu");

//创建输出流对象

FileWriter fw = **new** FileWriter("e.txt");

//void store(Writer writer, String comments)

prop.store(fw, "hello world");

//释放资源

fw.close();

}

**private** **static** **void** method2() **throws** FileNotFoundException, IOException {

//创建属性列表对象

Properties prop = **new** Properties();

//创建一个输入流对象

FileReader fr = **new** FileReader("d.txt");

//void load(Reader reader)

prop.load(fr);

//释放资源

fr.close();

System.*out*.println(prop);

}

**private** **static** **void** method() **throws** FileNotFoundException {

//创建属性列表对象

Properties prop = **new** Properties();

//添加映射关系

prop.setProperty("CZBK001", "zhangsan");

prop.setProperty("CZBK002", "lisi");

prop.setProperty("CZBK003", "wangwu");

//创建打印流对象

PrintWriter out = **new** PrintWriter("d.txt");

//void list(PrintWriter out)

prop.list(out);

//释放资源

out.close();

}

}

# 编码表

## 编码表的概述

编码表：把计算机底层的二进制数据转换成我们能看到的字符

\* ASCII

\*

\* GB2312 --- GBK

\* Unicode 所有的字符都占2个字节

\* UTF-8 长度可变的码表

\*

\* ANSI：本地编码表 gbk

\* Java中的字符串默认使用的ANSI（gbk）

\*

\* 乱码：编码保持前后一致即可解决

## Java中字符串的编码

### 常用方法

构造方法（字节数组转字符串）：

String():初始化一个新创建的 String 对象，使其表示一个空字符序列

String(byte[] bytes) 使用平台的默认字符集解码指定的 byte 数组，构造一个新的 String

String(byte[] bytes, Charset charset) 通过使用指定的 charset 解码指定的 byte 数组，构造一个新的 String

成员方法（字符串转字节数组）

getBytes() 使用平台的默认字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中

getBytes(Charset charset) 使用给定的 charset 将此 String 编码到 byte 序列，并将结果存储到新的 byte 数组

### 案例代码十一

package com.itheima\_09;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

/\*

\* 编码表：把计算机底层的二进制数据转换成我们能看到的字符

\* ASCII

\*

\* GB2312 --- GBK

\* Unicode 所有的字符都占2个字节

\* UTF-8 长度可变的码表

\*

\* ANSI：本地编码表 gbk

\* Java中的字符串默认使用的ANSI（gbk）

\*

\* 乱码：编码保持前后一致即可解决

\*

\*/

public class EncoderDemo {

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

//method();

FileInputStream fis = new FileInputStream("a.txt");

byte[] bys = new byte[1024];

int len = fis.read(bys);

//System.out.println(new String(bys,0,len));

System.out.println(new String(bys,0,len,"UTF-8"));

}

private static void method() throws UnsupportedEncodingException, FileNotFoundException, IOException {

String s = "高薪就业";

//byte[] bys = s.getBytes();//通过默认编码转换成数组

byte[] bys = s.getBytes("UTF-8");

FileOutputStream fos = new FileOutputStream("a.txt");

fos.write(bys);

fos.close();

}

}

## 字符流中的编码

### 常见对象

InputStreamReader(InputStream in, CharsetDecoder dec) 创建使用给定字符集解码器的 InputStreamReader

OutputStreamWriter(OutputStream out, CharsetEncoder enc) 创建使用给定字符集编码器的 OutputStreamWriter

### 案例代码十二

package com.itheima\_09;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.UnsupportedEncodingException;

/\*

\* 字符流中的编码

\*

\* 字符流 = 字节流 + 编码

\*

\*/

public class EncoderDemo2 {

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

//method();

OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("b.txt"),"UTF-8");

//OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("b.txt"));

String s = "迎娶白富美";

osw.write(s);

osw.close();

}

private static void method() throws IOException, UnsupportedEncodingException {

FileWriter fw = new FileWriter("b.txt");

String s = "月薪过万";

byte[] bys = s.getBytes("UTF-8");

//fw.write(s);

fw.write(new String(bys));

fw.close();

}

}