1.IO流案例

1.1集合到文件数据排序改进版【应用】

1.1.1案例需求

- 键盘录入5个学生信息(姓名,语文成绩,数学成绩,英语成绩)。要求按照成绩总分从高到低写入文本文件
- 格式：姓名,语文成绩,数学成绩,英语成绩 举例：林青霞,98,99,100

1.1.2分析步骤

1. 定义学生类
2. 创建TreeSet集合，通过比较器排序进行排序
3. 键盘录入学生数据
4. 创建学生对象，把键盘录入的数据对应赋值给学生对象的成员变量
5. 把学生对象添加到TreeSet集合
6. 创建字符缓冲输出流对象
7. 遍历集合，得到每一个学生对象
8. 把学生对象的数据拼接成指定格式的字符串
9. 调用字符缓冲输出流对象的方法写数据
10. 释放资源

1.1.3代码实现

- 学生类

```
public class Student {
    // 姓名
    private String name;
    // 语文成绩
    private int chinese;
    // 数学成绩
    private int math;
    // 英语成绩
    private int english;

    public Student() {
        super();
    }

    public Student(String name, int chinese, int math, int english) {
        super();
        this.name = name;
        this.chinese = chinese;
        this.math = math;
        this.english = english;
    }

    public String getName() {
        return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getChinese() {
        return chinese;
    }

    public void setChinese(int chinese) {
        this.chinese = chinese;
    }

    public int getMath() {
        return math;
    }

    public void setMath(int math) {
        this.math = math;
    }

    public int getEnglish() {
        return english;
    }

    public void setEnglish(int english) {
        this.english = english;
    }

    public int getSum() {
        return this.chinese + this.math + this.english;
    }
}
```

- 测试类

```
public class TreeSetToFileDemo {
    public static void main(String[] args) throws IOException {
        //创建TreeSet集合，通过比较器排序进行排序
        TreeSet<Student> ts = new TreeSet<Student>(new Comparator<Student>() {
            @Override
            public int compare(Student s1, Student s2) {
                //成绩总分从高到低
                int num = s2.getSum() - s1.getSum();
                //次要条件
                int num2 = num == 0 ? s1.getChinese() - s2.getChinese() : num;
                int num3 = num2 == 0 ? s1.getMath() - s2.getMath() : num2;
                int num4 = num3 == 0 ? s1.getName().compareTo(s2.getName()) :
num3;
                return num4;
            }
        });
```

```
16
17          //键盘录入学生数据
18          for (int i = 0; i < 5; i++) {
19              Scanner sc = new Scanner(System.in);
20              System.out.println("请录入第" + (i + 1) + "个学生信息：");
21              System.out.println("姓名：");
22              String name = sc.nextLine();
23              System.out.println("语文成绩：");
24              int chinese = sc.nextInt();
25              System.out.println("数学成绩：");
26              int math = sc.nextInt();
27              System.out.println("英语成绩：");
28              int english = sc.nextInt();
29
30              //创建学生对象，把键盘录入的数据对应赋值给学生对象的成员变量
31              Student s = new Student();
32              s.setName(name);
33              s.setChinese(chinese);
34              s.setMath(math);
35              s.setEnglish(english);
36
37              //把学生对象添加到TreeSet集合
38              ts.add(s);
39          }
40
41          //创建字符缓冲输出流对象
42          BufferedWriter bw = new BufferedWriter(new
    FileWriter("myCharStream\\ts.txt"));
43
44          //遍历集合，得到每一个学生对象
45          for (Student s : ts) {
46              //把学生对象的数据拼接成指定格式的字符串
47              //格式：姓名,语文成绩,数学成绩,英语成绩
48              StringBuilder sb = new StringBuilder();
49
     sb.append(s.getName()).append(",").append(s.getChinese()).append(",").append(s
    .getMath()).append(",").append(s.getEnglish()).append(",").append(s.getSum());
50
51  //              调用字符缓冲输出流对象的方法写数据
52              bw.write(sb.toString());
53              bw.newLine();
54              bw.flush();
55          }
56
57          //释放资源
58          bw.close();
59      }
60  }
```

1.2复制单级文件夹【应用】

1.2.1案例需求

- 把“E:\itcast”这个文件夹复制到模块目录下

1.2.2分析步骤

1. 创建数据源目录File对象，路径是E:\itcast
2. 获取数据源目录File对象的名称
3. 创建目的地目录File对象，路径由(模块名+第2步获取的名称)组成
4. 判断第3步创建的File是否存在，如果不存在，就创建
5. 获取数据源目录下所有文件的File数组
6. 遍历File数组，得到每一个File对象，该File对象，其实就是数据源文件
7. 获取数据源文件File对象的名称
8. 创建目的地文件File对象，路径由(目的地目录+第7步获取的名称)组成
9. 复制文件

   由于不清楚数据源目录下的文件都是什么类型的，所以采用字节流复制文件

   采用参数为File的构造方法

1.2.3代码实现

```
public class CopyFolderDemo {
    public static void main(String[] args) throws IOException {
        //创建数据源目录File对象，路径是E:\\itcast
        File srcFolder = new File("E:\\itcast");

        //获取数据源目录File对象的名称(itcast)
        String srcFolderName = srcFolder.getName();

        //创建目的地目录File对象，路径名是模块名+itcast组成(myCharStream\\itcast)
        File destFolder = new File("myCharStream",srcFolderName);

        //判断目的地目录对应的File是否存在，如果不存在，就创建
        if(!destFolder.exists()) {
            destFolder.mkdir();
        }

        //获取数据源目录下所有文件的File数组
        File[] listFiles = srcFolder.listFiles();

        //遍历File数组，得到每一个File对象，该File对象，其实就是数据源文件
        for(File srcFile : listFiles) {
            //数据源文件：E:\\itcast\\mn.jpg
            //获取数据源文件File对象的名称(mn.jpg)
            String srcFileName = srcFile.getName();
            //创建目的地文件File对象，路径名是目的地目录+mn.jpg组成
(myCharStream\\itcast\\mn.jpg)
            File destFile = new File(destFolder,srcFileName);
            //复制文件
            copyFile(srcFile,destFile);
        }
    }
```

```
31
32      private static void copyFile(File srcFile, File destFile) throws IOException {
33          BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(srcFile));
34          BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(destFile));
35
36          byte[] bys = new byte[1024];
37          int len;
38          while ((len=bis.read(bys))!=-1) {
39              bos.write(bys,0,len);
40          }
41
42          bos.close();
43          bis.close();
44      }
45  }
```

1.3复制多级文件夹【应用】

1.3.1案例需求

- 把“E:\itcast”这个文件夹复制到 F盘目录下

1.3.2分析步骤

1. 创建数据源File对象，路径是E:\itcast
2. 创建目的地File对象，路径是F:\
3. 写方法实现文件夹的复制，参数为数据源File对象和目的地File对象
4. 判断数据源File是否是文件

   是文件：直接复制，用字节流

   不是文件：

   在目的地下创建该目录

   遍历获取该目录下的所有文件的File数组，得到每一个File对象

   回到3继续(递归)

1.3.3代码实现

```
1   public class CopyFoldersDemo {
2       public static void main(String[] args) throws IOException {
3           //创建数据源File对象，路径是E:\\itcast
4           File srcFile = new File("E:\\itcast");
5           //创建目的地File对象，路径是F:\\
6           File destFile = new File("F:\\");
7
8           //写方法实现文件夹的复制，参数为数据源File对象和目的地File对象
9           copyFolder(srcFile,destFile);
10      }
11
```

```
    //复制文件夹
    private static void copyFolder(File srcFile, File destFile) throws IOException
{
        //判断数据源File是否是目录
        if(srcFile.isDirectory()) {
            //在目的地下创建和数据源File名称一样的目录
            String srcFileName = srcFile.getName();
            File newFolder = new File(destFile,srcFileName); //F:\\itcast
            if(!newFolder.exists()) {
                newFolder.mkdir();
            }

            //获取数据源File下所有文件或者目录的File数组
            File[] fileArray = srcFile.listFiles();

            //遍历该File数组，得到每一个File对象
            for(File file : fileArray) {
                //把该File作为数据源File对象，递归调用复制文件夹的方法
                copyFolder(file,newFolder);
            }
        } else {
            //说明是文件，直接复制，用字节流
            File newFile = new File(destFile,srcFile.getName());
            copyFile(srcFile,newFile);
        }
    }

    //字节缓冲流复制文件
    private static void copyFile(File srcFile, File destFile) throws IOException {
        BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(srcFile));
        BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(destFile));

        byte[] bys = new byte[1024];
        int len;
        while ((len = bis.read(bys)) != -1) {
            bos.write(bys, 0, len);
        }

        bos.close();
        bis.close();
    }
}
```

1.4复制文件的异常处理【应用】

1.4.1基本做法

```
public class CopyFileDemo {
    public static void main(String[] args) {

```

```
    }

    //try...catch...finally
    private static void method2() {
        FileReader fr = null;
        FileWriter fw = null;
        try {
            fr = new FileReader("fr.txt");
            fw = new FileWriter("fw.txt");

            char[] chs = new char[1024];
            int len;
            while ((len = fr.read()) != -1) {
                fw.write(chs, 0, len);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if(fw!=null) {
                try {
                    fw.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            if(fr!=null) {
                try {
                    fr.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    //抛出处理
    private static void method1() throws IOException {
        FileReader fr = new FileReader("fr.txt");
        FileWriter fw = new FileWriter("fw.txt");

        char[] chs = new char[1024];
        int len;
        while ((len = fr.read()) != -1) {
            fw.write(chs, 0, len);
        }

        fw.close();
        fr.close();
    }
}
```

1.4.2JDK7版本改进

```java
public class CopyFileDemo {
    public static void main(String[] args) {

    }

    //JDK7的改进方案
    private static void method3() {
        try(FileReader fr = new FileReader("fr.txt");
            FileWriter fw = new FileWriter("fw.txt");){
            char[] chs = new char[1024];
            int len;
            while ((len = fr.read()) != -1) {
                fw.write(chs, 0, len);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

1.4.3JDK9版本改进

```java
public class CopyFileDemo {
    public static void main(String[] args) {

    }

    //JDK9的改进方案
    private static void method4() throws IOException {
        FileReader fr = new FileReader("fr.txt");
        FileWriter fw = new FileWriter("fw.txt");
        try(fr;fw){
            char[] chs = new char[1024];
            int len;
            while ((len = fr.read()) != -1) {
                fw.write(chs, 0, len);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2.IO特殊操作流

2.1标准输入流【应用】

- System类中有两个静态的成员变量
  - public static final InputStream in：标准输入流。通常该流对应于键盘输入或由主机环境或用户指定的另一个输入源

    - public static final PrintStream out：标准输出流。通常该流对应于显示输出或由主机环境或用户指定的另一个输出目标
- 自己实现键盘录入数据

```
public class SystemInDemo {
    public static void main(String[] args) throws IOException {
        //public static final InputStream in：标准输入流
//        InputStream is = System.in;

//        int by;
//        while ((by=is.read())!=-1) {
//            System.out.print((char)by);
//        }

        //如何把字节流转换为字符流？用转换流
//        InputStreamReader isr = new InputStreamReader(is);
//        //使用字符流能不能够实现一次读取一行数据呢？可以
//        //但是，一次读取一行数据的方法是字符缓冲输入流的特有方法
//        BufferedReader br = new BufferedReader(isr);

        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("请输入一个字符串：");
        String line = br.readLine();
        System.out.println("你输入的字符串是：" + line);

        System.out.println("请输入一个整数：");
        int i = Integer.parseInt(br.readLine());
        System.out.println("你输入的整数是：" + i);

        //自己实现键盘录入数据太麻烦了，所以Java就提供了一个类供我们使用
        Scanner sc = new Scanner(System.in);
    }
}
```

2.2标准输出流【应用】

- System类中有两个静态的成员变量
    - public static final InputStream in：标准输入流。通常该流对应于键盘输入或由主机环境或用户指定的另一个输入源
    - public static final PrintStream out：标准输出流。通常该流对应于显示输出或由主机环境或用户指定的另一个输出目标
- 输出语句的本质：是一个标准的输出流
    - PrintStream ps = System.out;
    - PrintStream类有的方法，System.out都可以使用
- 示例代码

```
public class SystemOutDemo {
    public static void main(String[] args) {
```

```
        //public static final PrintStream out：标准输出流
        PrintStream ps = System.out;

        //能够方便地打印各种数据值
//        ps.print("hello");
//        ps.print(100);

//        ps.println("hello");
//        ps.println(100);

        //System.out的本质是一个字节输出流
        System.out.println("hello");
        System.out.println(100);

        System.out.println();
//        System.out.print();
    }
}
```

2.3字节打印流【应用】

- 打印流分类
  - 字节打印流：PrintStream
  - 字符打印流：PrintWriter
- 打印流的特点
  - 只负责输出数据，不负责读取数据
  - 永远不会抛出IOException
  - 有自己的特有方法
- 字节打印流
  - PrintStream(String fileName)：使用指定的文件名创建新的打印流
  - 使用继承父类的方法写数据，查看的时候会转码；使用自己的特有方法写数据，查看的数据原样输出
  - 可以改变输出语句的目的地

    public static void setOut(PrintStream out)：重新分配“标准”输出流
- 示例代码

```
public class PrintStreamDemo {
    public static void main(String[] args) throws IOException {
        //PrintStream(String fileName)：使用指定的文件名创建新的打印流
        PrintStream ps = new PrintStream("myOtherStream\\ps.txt");

        //写数据
        //字节输出流有的方法
//        ps.write(97);

        //使用特有方法写数据
//        ps.print(97);
//        ps.println();
//        ps.print(98);
        ps.println(97);
```

```
        ps.println(98);

        //释放资源
        ps.close();
    }
}
```

2.4字符打印流【应用】

- 字符打印流构造房方法

方法名	说明
PrintWriter(String fileName)	使用指定的文件名创建一个新的PrintWriter，而不需要自动执行刷新
PrintWriter(Writer out, boolean autoFlush)	创建一个新的PrintWriter out：字符输出流 autoFlush： 一个布尔值，如果为真，则println ， printf ，或format方法将刷新输出缓冲区

- 示例代码

```
public class PrintWriterDemo {
    public static void main(String[] args) throws IOException {
        //PrintWriter(String fileName) ：使用指定的文件名创建一个新的PrintWriter，而不需要自动执行行刷新
//        PrintWriter pw = new PrintWriter("myOtherStream\\pw.txt");

//        pw.write("hello");
//        pw.write("\r\n");
//        pw.flush();
//        pw.write("world");
//        pw.write("\r\n");
//        pw.flush();

//        pw.println("hello");
        /*
            pw.write("hello");
            pw.write("\r\n");
         */
//        pw.flush();
//        pw.println("world");
//        pw.flush();

        //PrintWriter(Writer out, boolean autoFlush)：创建一个新的PrintWriter
        PrintWriter pw = new PrintWriter(new FileWriter("myOtherStream\\pw.txt"),true);
//        PrintWriter pw = new PrintWriter(new FileWriter("myOtherStream\\pw.txt"),false);

        pw.println("hello");
        /*
```

```
            pw.write("hello");
            pw.write("\r\n");
            pw.flush();
         */
        pw.println("world");

        pw.close();
    }
}
```

2.5复制Java文件打印流改进版【应用】

- 案例需求
  - 把模块目录下的PrintStreamDemo.java 复制到模块目录下的 Copy.java
- 分析步骤
  - 根据数据源创建字符输入流对象
  - 根据目的地创建字符输出流对象
  - 读写数据，复制文件
  - 释放资源
- 代码实现

```
public class CopyJavaDemo {
    public static void main(String[] args) throws IOException {
        /*
        //根据数据源创建字符输入流对象
        BufferedReader br = new BufferedReader(new
FileReader("myOtherStream\\PrintStreamDemo.java"));
        //根据目的地创建字符输出流对象
        BufferedWriter bw = new BufferedWriter(new
FileWriter("myOtherStream\\Copy.java"));

        //读写数据，复制文件
        String line;
        while ((line=br.readLine())!=null) {
            bw.write(line);
            bw.newLine();
            bw.flush();
        }

        //释放资源
        bw.close();
        br.close();
        */

        //根据数据源创建字符输入流对象
        BufferedReader br = new BufferedReader(new
FileReader("myOtherStream\\PrintStreamDemo.java"));
        //根据目的地创建字符输出流对象
        PrintWriter pw = new PrintWriter(new
FileWriter("myOtherStream\\Copy.java"),true);

```

```
        //读写数据，复制文件
        String line;
        while ((line=br.readLine())!=null) {
            pw.println(line);
        }

        //释放资源
        pw.close();
        br.close();
    }
}
```

2.6对象序列化流【应用】

- 对象序列化介绍
  - 对象序列化：就是将对象保存到磁盘中，或者在网络中传输对象
  - 这种机制就是使用一个字节序列表示一个对象，该字节序列包含：对象的类型、对象的数据和对象中存储的属性等信息
  - 字节序列写到文件之后，相当于文件中持久保存了一个对象的信息
  - 反之，该字节序列还可以从文件中读取回来，重构对象，对它进行反序列化
- 对象序列化流： ObjectOutputStream
  - 将Java对象的原始数据类型和图形写入OutputStream。 可以使用ObjectInputStream读取（重构）对象。 可以通过使用流的文件来实现对象的持久存储。 如果流是网络套接字流，则可以在另一个主机上或另一个进程中重构对象
- 构造方法

方法名	说明
ObjectOutputStream(OutputStream out)	创建一个写入指定的OutputStream的ObjectOutputStream

- 序列化对象的方法

方法名	说明
void writeObject(Object obj)	将指定的对象写入ObjectOutputStream

- 示例代码
  - 学生类

```
public class Student implements Serializable {
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
```

```
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

- 测试类

```
public class ObjectOutputStreamDemo {
    public static void main(String[] args) throws IOException {
        //ObjectOutputStream(OutputStream out)：创建一个写入指定的OutputStream的ObjectOutputStream
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("myOtherStream\\oos.txt"));

        //创建对象
        Student s = new Student("林青霞",30);

        //void writeObject(Object obj)：将指定的对象写入ObjectOutputStream
        oos.writeObject(s);

        //释放资源
        oos.close();
    }
}
```

- 注意事项
  - 一个对象要想被序列化，该对象所属的类必须必须实现Serializable 接口
  - Serializable是一个标记接口，实现该接口，不需要重写任何方法

2.7对象反序列化流【应用】

- 对象反序列化流： ObjectInputStream
  - ObjectInputStream反序列化先前使用ObjectOutputStream编写的原始数据和对象
- 构造方法

方法名	说明
ObjectInputStream(InputStream in)	创建从指定的InputStream读取的ObjectInputStream

- 反序列化对象的方法

方法名	说明
Object readObject()	从ObjectInputStream读取一个对象

- 示例代码

```
public class ObjectInputStreamDemo {
    public static void main(String[] args) throws IOException,
ClassNotFoundException {
        //ObjectInputStream(InputStream in)：创建从指定的InputStream读取的
ObjectInputStream
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("myOtherStream\\oos.txt"));

        //Object readObject()：从ObjectInputStream读取一个对象
        Object obj = ois.readObject();

        Student s = (Student) obj;
        System.out.println(s.getName() + "," + s.getAge());

        ois.close();
    }
}
```

2.8serialVersionUID&transient【应用】

- serialVersionUID
  - 用对象序列化流序列化了一个对象后，假如我们修改了对象所属的类文件，读取数据会不会出问题呢？
    - 会出问题，会抛出InvalidClassException异常
  - 如果出问题了，如何解决呢？
    - 重新序列化
    - 给对象所属的类加一个serialVersionUID
      - private static final long serialVersionUID = 42L;
- transient
  - 如果一个对象中的某个成员变量的值不想被序列化，又该如何实现呢？

    - 给该成员变量加transient关键字修饰，该关键字标记的成员变量不参与序列化过程
- 示例代码
    - 学生类

```
public class Student implements Serializable {
    private static final long serialVersionUID = 42L;
    private String name;
//    private int age;
    private transient int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

//    @Override
//    public String toString() {
//        return "Student{" +
//                "name='" + name + '\'' +
//                ", age=" + age +
//                '}';
//    }
}
```

    - 测试类

```
public class ObjectStreamDemo {
    public static void main(String[] args) throws IOException,
ClassNotFoundException {
//        write();
        read();
    }

```

```
    //反序列化
    private static void read() throws IOException, ClassNotFoundException {
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("myOtherStream\\oos.txt"));
        Object obj = ois.readObject();
        Student s = (Student) obj;
        System.out.println(s.getName() + "," + s.getAge());
        ois.close();
    }

    //序列化
    private static void write() throws IOException {
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("myOtherStream\\oos.txt"));
        Student s = new Student("林青霞", 30);
        oos.writeObject(s);
        oos.close();
    }
}
```

3.Properties集合

3.1Properties作为Map集合的使用【应用】

- Properties介绍
  - 是一个Map体系的集合类
  - Properties可以保存到流中或从流中加载
  - 属性列表中的每个键及其对应的值都是一个字符串
- Properties基本使用

```
public class PropertiesDemo01 {
    public static void main(String[] args) {
        //创建集合对象
//        Properties<String,String> prop = new Properties<String,String>(); //错
误
        Properties prop = new Properties();

        //存储元素
        prop.put("itheima001", "林青霞");
        prop.put("itheima002", "张曼玉");
        prop.put("itheima003", "王祖贤");

        //遍历集合
        Set<Object> keySet = prop.keySet();
        for (Object key : keySet) {
            Object value = prop.get(key);
            System.out.println(key + "," + value);
        }
    }
}
```

3.2Properties作为Map集合的特有方法【应用】

- 特有方法

方法名	说明
Object setProperty(String key, String value)	设置集合的键和值，都是String类型，底层调用 Hashtable方法 put
String getProperty(String key)	使用此属性列表中指定的键搜索属性
Set stringPropertyNames()	从该属性列表中返回一个不可修改的键集，其中键及其对应的值是字符串

- 示例代码

```
public class PropertiesDemo02 {
    public static void main(String[] args) {
        //创建集合对象
        Properties prop = new Properties();

        //Object setProperty(String key, String value)：设置集合的键和值，都是String类型，底层调用Hashtable方法put
        prop.setProperty("itheima001", "林青霞");
        /*
            Object setProperty(String key, String value) {
                return put(key, value);
            }

            Object put(Object key, Object value) {
                return map.put(key, value);
            }
         */
        prop.setProperty("itheima002", "张曼玉");
        prop.setProperty("itheima003", "王祖贤");

        //String getProperty(String key)：使用此属性列表中指定的键搜索属性
//        System.out.println(prop.getProperty("itheima001"));
//        System.out.println(prop.getProperty("itheima0011"));

//        System.out.println(prop);

        //Set<String> stringPropertyNames()：从该属性列表中返回一个不可修改的键集，其中键及其对应的值是字符串
        Set<String> names = prop.stringPropertyNames();
        for (String key : names) {
//            System.out.println(key);
            String value = prop.getProperty(key);
            System.out.println(key + "," + value);
        }
    }
}
```

3.3Properties和IO流相结合的方法【应用】

- 和IO流结合的方法

方法名	说明
void load(InputStream inStream)	从输入字节流读取属性列表（键和元素对）
void load(Reader reader)	从输入字符流读取属性列表（键和元素对）
void store(OutputStream out, String comments)	将此属性列表（键和元素对）写入此 Properties表中，以适合于使用load(InputStream)方法的格式写入输出字节流
void store(Writer writer, String comments)	将此属性列表（键和元素对）写入此 Properties表中，以适合使用load(Reader)方法的格式写入输出字符流

- 示例代码

```
public class PropertiesDemo03 {
    public static void main(String[] args) throws IOException {
        //把集合中的数据保存到文件
//        myStore();

        //把文件中的数据加载到集合
        myLoad();

    }

    private static void myLoad() throws IOException {
        Properties prop = new Properties();

        //void load(Reader reader)：
        FileReader fr = new FileReader("myOtherStream\\fw.txt");
        prop.load(fr);
        fr.close();

        System.out.println(prop);
    }

    private static void myStore() throws IOException {
        Properties prop = new Properties();

        prop.setProperty("itheima001","林青霞");
        prop.setProperty("itheima002","张曼玉");
        prop.setProperty("itheima003","王祖贤");

        //void store(Writer writer, String comments)：
        FileWriter fw = new FileWriter("myOtherStream\\fw.txt");
        prop.store(fw,null);
        fw.close();
    }
```

```
}
```

3.4游戏次数案例【应用】

- 案例需求
  - 实现猜数字小游戏只能试玩3次，如果还想玩，提示：游戏试玩已结束，想玩请充值(www.itcast.cn)
- 分析步骤
  1. 写一个游戏类，里面有一个猜数字的小游戏
  2. 写一个测试类，测试类中有main()方法，main()方法中写如下代码：

     从文件中读取数据到Properties集合，用load()方法实现

     文件已经存在：game.txt

     里面有一个数据值：count=0

     通过Properties集合获取到玩游戏的次数

     判断次数是否到到3次了

     如果到了，给出提示：游戏试玩已结束，想玩请充值(www.itcast.cn)

     如果不到3次：

     次数+1，重新写回文件，用Properties的store()方法实现玩游戏
- 代码实现

```
public class PropertiesTest {
    public static void main(String[] args) throws IOException {
        //从文件中读取数据到Properties集合，用load()方法实现
        Properties prop = new Properties();

        FileReader fr = new FileReader("myOtherStream\\game.txt");
        prop.load(fr);
        fr.close();

        //通过Properties集合获取到玩游戏的次数
        String count = prop.getProperty("count");
        int number = Integer.parseInt(count);

        //判断次数是否到到3次了
        if(number >= 3) {
            //如果到了，给出提示：游戏试玩已结束，想玩请充值(www.itcast.cn)
            System.out.println("游戏试玩已结束，想玩请充值(www.itcast.cn)");
        } else {
            //玩游戏
            GuessNumber.start();

            //次数+1，重新写回文件，用Properties的store()方法实现
            number++;
            prop.setProperty("count",String.valueOf(number));
            FileWriter fw = new FileWriter("myOtherStream\\game.txt");
            prop.store(fw,null);
            fw.close();
        }
```

```
29      }
30  }
```