【济南中心】JAVA 编程阶梯：基础篇之第二十二章

IO 流之序列流：

1.什么是序列流

* 序列流可以把多个字节输入流整合成一个，从序列流中读取数据时，将从被整合的第一个流开始读，读完一个之后继续读第二个，以此类推.

2.使用方式

* 整合两个: SequenceInputStream(InputStream, InputStream)

```
FileInputStream fis1 = new FileInputStream("a.txt");                    //创建输入流对象,关联 a.txt
                        FileInputStream fis2 = new FileInputStream("b.txt");                    //创建输入流对象,关联 b.txt
                        SequenceInputStream sis = new SequenceInputStream(fis1, fis2);      //将两个流整合成一个流
                        FileOutputStream fos = new FileOutputStream("c.txt");                   //创建输出流对象,关联 c.txt

                        int b;
                        while((b = sis.read()) != -1)                                                   {                                                                                       //用整合后的读
                                fos.write(b);
                                                                //写到指定文件上
                        }

                        sis.close();
                        fos.close();
```

IO 流之序列流整合多个：

整合多个: SequenceInputStream(Enumeration)

```
FileInputStream fis1 = new FileInputStream("a.txt");        //创建输入流对象,关联 a.txt
                FileInputStream fis2 = new FileInputStream("b.txt");        //创建输入流对象,关联 b.txt
                FileInputStream fis3 = new FileInputStream("c.txt");        //创建输入流对象,关联 c.txt
                Vector<InputStream> v = new Vector<>();                                         //创建 vector 集合对象
```

```
v.add(fis1);
        //将流对象添加
v.add(fis2);
v.add(fis3);
Enumeration<InputStream> en = v.elements();                                    //
获取枚举引用
SequenceInputStream sis = new SequenceInputStream(en);       //传递给
SequenceInputStream 构造
FileOutputStream fos = new FileOutputStream("d.txt");
int b;
while((b = sis.read()) != -1) {
        fos.write(b);
}

sis.close();
fos.close();
```

IO 流之内存输出流：

1.什么是内存输出流

* 该输出流可以向内存中写数据, 把内存当作一个缓冲区, 写出之后可以一次性获取出所有数据

2.使用方式

* 创建对象: new ByteArrayOutputStream()
* 写出数据: write(int), write(byte[])
* 获取数据: toByteArray()

```
FileInputStream fis = new FileInputStream("a.txt");
                    ByteArrayOutputStream baos = new ByteArrayOutputStream();
                    int b;
                    while((b = fis.read()) != -1) {
                            baos.write(b);
                    }

                    //byte[] newArr =
baos.toByteArray();                                 //将内存缓冲区中所有的字节存储在 newArr 中
                    //System.out.println(new String(newArr));
                    System.out.println(baos);
```

```
                        fis.close();
```

例子：定义一个文件输入流,调用 read(byte[] b)方法,将 a.txt 文件中的内容打印出来(byte 数组大小限制为 5)

```
FileInputStream fis = new FileInputStream("a.txt");                                    //创建字节输入流,关联 a.txt
                        ByteArrayOutputStream baos = new ByteArrayOutputStream();                //创建内存输出流
                        byte[] arr = new byte[5];                                                                                //创建字节数组,大小为 5
                        int len;
                        while((len = fis.read(arr)) != -1){                                        //将文件上的数据读到字节数组中
                                baos.write(arr, 0, len);                                                        //将字节数组的数据写到内存缓冲区中
                        }
                        System.out.println(baos);
                                //将内存缓冲区的内容转换为字符串打印
                        fis.close();
```

IO 流之随机访问流概述和读写数据：

A:随机访问流概述

* RandomAccessFile 概述

* RandomAccessFile 类不属于流，是 Object 类的子类。但它融合了 InputStream 和 OutputStream 的功能。

* 支持对随机访问文件的读取和写入。

B:read(),write(),seek()

IO 流之对象操作流 ObjecOutputStream:

1.什么是对象操作流

* 该流可以将一个对象写出, 或者读取一个对象到程序中. 也就是执行了序列化和反序列化的操作.

2.使用方式

* 写出: new ObjectOutputStream(OutputStream), writeObject()

```
public class Demo3_ObjectOutputStream {

	/**
	 * @param args
	 * @throws IOException
	 * 将对象写出,序列化
	 */
	public static void main(String[] args) throws IOException {
		Person p1 = new Person("张三", 23);
		Person p2 = new Person("李四", 24);
//		 FileOutputStream fos = new FileOutputStream("e.txt");
//		 fos.write(p1);
//		 FileWriter fw = new FileWriter("e.txt");
//		 fw.write(p1);
		//无论是字节输出流,还是字符输出流都不能直接写出对象
		ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("e.txt"));//创建对象输出流
		oos.writeObject(p1);
		oos.writeObject(p2);
		oos.close();
	}

}
```

IO 流之对象操作流 ObjectInputStream:

读取: new ObjectInputStream(InputStream), readObject()

```
public class Demo3_ObjectInputStream {

	/**
	 * @param args
	 * @throws IOException
	 * @throws ClassNotFoundException
	 * @throws FileNotFoundException
	 * 读取对象,反序列化
	 */
	public static void main(String[] args) throws IOException,
ClassNotFoundException {
		ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("e.txt"));
		Person p1 = (Person) ois.readObject();
```

```
                Person p2 = (Person) ois.readObject();
                System.out.println(p1);
                System.out.println(p2);
                ois.close();
        }

    }
```

IO流之对象操作流优化:

将对象存储在集合中写出

```
Person p1 = new Person("张三", 23);
        Person p2 = new Person("李四", 24);
        Person p3 = new Person("马哥", 18);
        Person p4 = new Person("辉哥", 20);

        ArrayList<Person> list = new ArrayList<>();
        list.add(p1);
        list.add(p2);
        list.add(p3);
        list.add(p4);

        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("f.txt"));
        oos.writeObject(list);
         //写出集合对象

        oos.close();
```

读取到的是一个集合对象

```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("f.txt"));
                ArrayList<Person> list =
(ArrayList<Person>)ois.readObject();        //泛型在运行期会被擦除,索引运行期相当于没有泛型
                //想去掉黄色可以加注解
                @SuppressWarnings("unchecked")
                for (Person person : list) {
                        System.out.println(person);
                }

            ois.close();
```

数据输入输出流:

1.什么是数据输入输出流

* DataInputStream, DataOutputStream 可以按照基本数据类型大小读写数据

* 例如按Long大小写出一个数字, 写出时该数据占8字节. 读取的时候也可以按照 Long 类型读取, 一次读取 8 个字节.

2.使用方式

* DataOutputStream(OutputStream), writeInt(), writeLong()

```
DataOutputStream dos = new DataOutputStream(new FileOutputStream("b.txt"));
                        dos.writeInt(997);
                        dos.writeInt(998);
                        dos.writeInt(999);

                        dos.close();
```

* DataInputStream(InputStream), readInt(), readLong()

```
DataInputStream dis = new DataInputStream(new FileInputStream("b.txt"));
                        int x = dis.readInt();
                        int y = dis.readInt();
                        int z = dis.readInt();
                        System.out.println(x);
                        System.out.println(y);
                        System.out.println(z);
                        dis.close();
```

打印流的概述和特点：

1.什么是打印流

* 该流可以很方便的将对象的 toString()结果输出, 并且自动加上换行, 而且可以使用自动刷出的模式

* System.out 就是一个 PrintStream, 其默认向控制台输出信息

```
PrintStream ps = System.out;
                        ps.println(97);                                         //其实底层用的是 Integer.toString(x),将 x 转换为数字字符串打印
                        ps.println("xxx");
                        ps.println(new Person("张三", 23));
                        Person p = null;
                        ps.println(p);                                          //如果是null,就返回 null,如果不是 null,就调用对象的 toString()
```

2. 使用方式

* 打印: print(), println()

* 自动刷出: PrintWriter(OutputStream out, boolean autoFlush, String encoding)

* 打印流只操作数据目的

```
PrintWriter pw = new PrintWriter(new FileOutputStream("g.txt"), true);
                        pw.write(97);
                        pw.print("大家好");
                        pw.println("你好");                                    //自动刷出,只针对的是 println 方法
                        pw.close();
```

IO 流之标准输入输出流概述和输出语句：

* 1.什么是标准输入输出流(掌握)

* System.in 是 InputStream, 标准输入流, 默认可以从键盘输入读取字节数据

* System.out 是 PrintStream, 标准输出流, 默认可以向 Console 中输出字符和字节数据

* 2.修改标准输入输出流(了解)

* 修改输入流: System.setIn(InputStream)

* 修改输出流: System.setOut(PrintStream)

```
System.setIn(new FileInputStream("a.txt"));                                 //修改标准输入流
                        System.setOut(new PrintStream("b.txt"));                                    //修改标准输出流

                        InputStream in = System.in;                                                 //获取标准输入流
                        PrintStream ps = System.out;                                                //获取标准输出流
                        int b;
                        while((b = in.read()) != -1)
{                                                                           //从 a.txt 上读取数据
```

```
                ps.write(b);
                //将数据写到 b.txt 上
            }

            in.close();
            ps.close();
```

IO 流之修改标准输入输出流拷贝图片：

```
System.setIn(new FileInputStream("IO 图片.png"));                    //改变标准输入流
        System.setOut(new PrintStream("copy.png"));                          //改变标准输出流

        InputStream is =
System.in;                                                          //获取标准输入流
        PrintStream ps = System.out;                                                    //
获取标准输出流

        int len;
        byte[] arr = new byte[1024 * 8];

        while((len = is.read(arr)) != -1) {
                ps.write(arr, 0, len);
        }

        is.close();
        ps.close();
```

IO 流之两种方式实现键盘录入(了解)：

A:BufferedReader 的 readLine 方法。

* BufferedReader br = new BufferedReader(new

InputStreamReader(System.in));

B:Scanner

IO 流之 Properties 的概述和作为 Map 集合的使用(了解)

A:Properties 的概述

* 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。

IO 流之 Properties 的特殊功能使用(了解)

A:Properties 的特殊功能

* public Object setProperty(String key,String value)

* public String getProperty(String key)

* public Enumeration<String> stringPropertyNames()

IO 流之 Properties 的 load()和 store()功能(了解)

* A:Properties 的 load()和 store()功能

* B:案例演示

* Properties 的 load()和 store()功能

识别二维码 关注黑马程序员视频库
免费获得更多 IT 资源