IO流总结

一：IO流整体说明

用的是装饰模式，装饰模式有四个角色，分别是通用接口、本原、抽象装饰、具体装饰，因为是将本原聚合到了装饰中，所以可以将本原或装饰通过构造子注入到另一个装饰，多个装饰可以随意的组合，形成一个综合的功能对象。

io流分为输入流及输出流，输入流是文件到内存，输出流是内存到文件。

流分为低层流及高层流，在实例化高层流时必须通过构造子将低层流注入到高层流中。

**输入流通用的方法是：** read(), read(char[]), read(byte[]), readLine()

**输出流的通用方法是：**  write(int), write(byte[]), write(char[]), print("string")

当读取一个文件时，

对于以字节/字节数组,字符/字符数组读取的结束条件是!=-1,

对于以字符串读取的结束条件是!=null。(如readLine())

对于有缓存的输出时，要想写入到文件必须用flush()或close()。

二：输入字节流的体系结构

InputStream

FileInputStream FilterInputStream

ObjectInputStream DataInputStream BIS

实现代码:

//读一个字节

int temp=fis.read();

while(temp!=-1) {

System.out.print(temp);

temp=fis.read();

}

//读一批字节

byte[] bytes=new byte[1024];//1Kb

int result=0;

while((result=fis.read(bytes))!=-1){

String temp=new String(bytes,0,result);

}

三：输出的字节流体系结构

OutputStream

FileOutputStream FilterOutputStreame

ObjectOutputStream BufferedOutputStream DataOutputStream PrintStream

实现代码:

int temp;

fis.write(temp);

byte temps[]=new byte[1000];

fos.write(temps);

四：输入的字符流体系结构

Reader

FileReader InputStreamReader(将字节流转化字符流)

BufferedReader(有缓存)

实现代码：

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

List<String> list = new ArrayList<String>();

String temp=null;

while((temp=br.readLine())!=null){

list.add(temp);

}

return list.toArray(new String[]());

五：输出的字符流体系结构?

Writer

FileWriter OutputStreamWriter

BufferedWriter PrintWriter

实现代码：

fw.write(int);

fw.write(char[]);

fw.write("string");

六：PrintStream、BufferedWriter、PrintWriter的比较

PrintStream:属于字节流，即可以打印字节也可以打印数组

写入文件不用close或flush

e.printStackTrace()、System.out.println()都是它的实例。

BufferedWriter、PrintWriter:属于字符流，只可以打印字符或字符串

因为有缓存，要想写入文件必须close或flush

来吧，同志们，为复习网络编程做准备……

一.理论准备

        流是个抽象的概念，是**对输入输出设备的抽象**，Java程序中，对于数据的输入/输出操作都是以“流”的方式进行，设备可以是文件、网络、内存等。流具有**方向性**，至于是输入流还是输出流则是一个相对的概念，一般以程序(小马哥说的是机器)为参考，如果数据的流向是程序至设备，我们成为输出流，反之我们称为输入流，可以将流想象成一个“水流管道”(很多资料都这么讲的)，自然就出现了方向的概念。

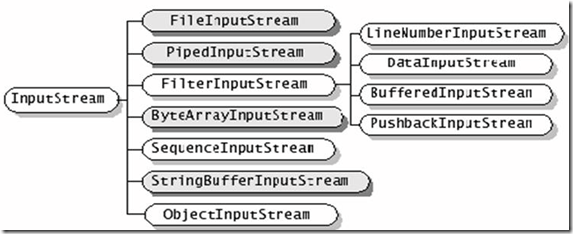
        流把I/O设备内部的具体操作给隐藏起来了。所有**InputStream和Reader**的派生类都有一个基本的，继承下来的，能读取单个或byte数组的**read( )**方法。

        Java分为**字节流（Stream结尾）和字符流（Reader、Write结尾）**，再分为输入流（InputStream、Reader）和输出流 （OutputStream、Write），输入输出相对于内存而言。在读字符的时候用字符流，如文本文件、XML(我想xml明明是字母字符组成的，属 于ASCII文件，为何不用stream读取呢？)等。在读二进制文件时候用字节流，如RAR、EXE等不是文本以外的文件(图片)。Buffered开头的流只是加了缓冲区，为了读写提高效率。**字符流不能直接输出，需要转换成字节流才能输出**(这个确实是刚知道的)！

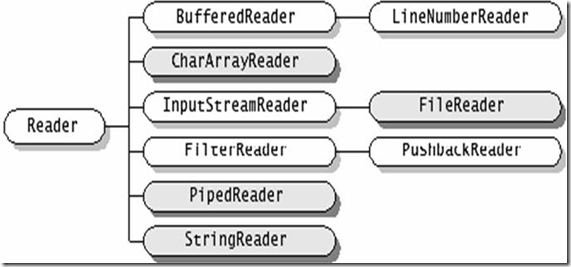
        Java 2 SDK中有三种基本类型的节点：文件(file)、内存(memory)、管道(pipe)。

下面来看郑莉教材上IO章节的那个经典图片。

        继承自InputStream/OutputStream的流都是用于向程序中输入/输出数据，且数据的单位都是字节(byte=8bit)，如图，深色的为节点流，浅色的为处理流。

[](http://images.cnitblog.com/blog/387014/201309/05110932-0d7f84da4d2447269c1a31d7b5fdbd20.png)

          继承自Reader/Writer的流都是用于向程序中多次西方v输入/输出数据，且数据的单位都是字符(2byte=16bit)，如图，深色的为节点流，浅色的为处理流。

[](http://images.cnitblog.com/blog/387014/201309/05110933-a01fe3a5bbd340f58228de4e6786d7b0.png)

二.用法分析

        Java IO的一般使用原则(部分来自百度文库)：   
        (1) 按数据来源（去向）分类：

1. **是文件：** FileInputStream, FileOutputStream, FileReader, FileWriter
2. **是byte[ ]**：ByteArrayInputStream, ByteArrayOutputStream
3. **是Char[ ]:** CharArrayReader, CharArrayWriter
4. **是String:** StringBufferInputStream, StringReader, StringWriter
5. **网络数据流：**InputStream, OutputStream, Reader, Writer   
   (2) 按是否格式化输出分：   
   要格式化输出：PrintStream, PrintWriter   
   (3) 按是否要缓冲分：   
   要缓冲：BufferedInputStream, BufferedOutputStream,
6. BufferedReader, BufferedWriter。

       (4) 按数据格式分：

1. 二进制格式（只要不能确定是纯文本的）: InputStream, OutputStream及其所有带Stream结束的子类
2. 纯文本格式（含纯英文与汉字或其他编码方式）；Reader, Writer及其所有带Reader, Writer的子类   
   (5) 按输入输出分：   
   输入：Reader, InputStream类型的子类；输出：Writer, OutputStream类型的子类   
   (6) 特殊需要：
3. 从Stream到Reader,Writer的转换类：InputStreamReader, OutputStreamWriter
4. 对象输入输出：ObjectInputStream, ObjectOutputStream
5. 进程间通信：PipeInputStream, PipeOutputStream, PipeReader, PipeWriter
6. 合并输入：SequenceInputStream
7. 更特殊的需要：PushbackInputStream, PushbackReader, LineNumberInputStream, LineNumberReader

**(7) 决定使用哪个类以及它的构造进程的一般准则如下（不考虑特殊需要）：**

1. 考虑最原始的数据格式是什么？
2. 是否为文本、纯文本？
3. 是输入还是输出?
4. 是否需要转换流？
5. InputStreamReader, OutputStreamWriter?
6. 数据来源（去向）是什么：文件？内存？网络？
7. 是否要缓冲：bufferedReader （特别注明：一定要注意的是readLine()是否有定义，有什么比read, write更特殊的输入或输出方法）
8. 是否要格式化输出：print。

三.若干实例

         还是寒假时候写的，权当复习了，折叠代码的插件找不到了，先看着吧。

         1.System.in

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*

\* System.in是InputStream static final的，包含了多态,叫同步式或者阻塞式

\* 读取ASCII和二进制文件(图片)，而字母就是ASCII字符(个人理解)。

\*/

public class TestSystemIn {

public static void main(String[] args) {

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);//有readline

String s = null;

try {

s = br.readLine();

while(s!=null) {

if(s.equalsIgnoreCase("exit")) {

break;

}

System.out.println(s.toUpperCase());

s = br.readLine();

}

br.close();

}catch (IOException e) {

e.printStackTrace();

}

}

}

           2.buffer

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class TestBuffer {

public static void main(String[] args) {

try {

//查看修改日就可以判断文件是否是新建的了

BufferedWriter bw = new BufferedWriter(new FileWriter("d:/java.txt"));

BufferedReader br = new BufferedReader(new FileReader("d:/java.txt"));

String s = null;

for(int i=1; i<100; i++) {

s = String.valueOf(Math.random());

bw.write(s);

bw.newLine();//换行

}

//刷新该流的缓冲，br没有该方法

bw.flush();

while((s=br.readLine())!=null) {

System.out.println(s);

}

bw.close();

br.close();

}catch (IOException e) {

e.printStackTrace();

}

}

}

             3.FileInputStream

import java.io.\*;

public class TestFileInputStream {

public static void main(String[] args) {

FileInputStream in = null;

try {

in = new FileInputStream("e:/1.txt");

}catch(FileNotFoundException e) {

System.out.println("找不到文件");

System.exit(-1);

}

//下面表示找到了文件

int tag = 0;

try {

long num = 0;

while((tag = in.read())!=-1) {

//read是字节流，若是有汉字就显示不正常了,使用reader就解决了

System.out.print((char)tag);

num++;

}

in.close();

System.out.println();

System.out.println("共读取了" + num + "字符");

}catch(IOException e1) {//read和close都会抛出IOException

System.out.println("文件读取错误");

System.exit(-1);

}

}

}

              4.FileOutputStream实现复制功能

import java.io.\*;

/\*

\* 实现复制功能

\*/

public class TestFileOutputStream {

public static void main(String[] args) {

int b = 0;

FileInputStream in = null;

FileOutputStream out = null;

try {

in = new FileInputStream("d:/java.txt");

//下面的若是不存在的话会自动建立

out = new FileOutputStream("d:/my\_java.txt");

while((b=in.read())!=-1) {

out.write(b);

}

in.close();

out.close();

}catch(FileNotFoundException e) {

System.out.println("找不到指定文件");

System.exit(-1);

}catch(IOException e1) {

System.out.println("文件复制错误");

System.exit(-1);

}

System.out.println("文件已复制");

}

}

              5.ObjectOutputStream与Serializable

import java.io.\*;

/\*

\* transient（透明的），可以用来修饰成员变量，

\* 当进行序列化时不予考虑，修饰int 的话，不管原来的值是多少

\* 输出的就是0

\*/

public class TestObjectIO {

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

T t = new T();

t.k = 8;

FileOutputStream fos = new FileOutputStream("d:/1.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(t);

oos.flush();

oos.close();

FileInputStream fis = new FileInputStream("d:/1.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

T tRead = (T)ois.readObject();

System.out.println(tRead.i + " " + tRead.j + " " + tRead.k);

}

}

class T implements Serializable {

int i = 10;

int j = 9;

double d = 2.3;

int k = 15;

}

             6.转换编码方式

import java.io.\*;

/\*

\* 中文windows默认GBK编码方式

\* 追加的内容显示为问号，不知道咋回事

\*/

public class TestTransForm {

public static void main(String[] args) {

try {

OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("d:/java.txt"));

osw.write("你好123");//可以直接写入字符串，包括中文,因为外边的是字符流

System.out.println("编码方式：" + osw.getEncoding());//ISO8859\_1是西欧语言，又叫latin-1，此时未考虑东方人，国标（ISO）为Unicode

osw.close();

osw = new OutputStreamWriter(new FileOutputStream("d:/java.txt",true),"ISO8859\_1");//true表示追加

osw.write("这是追加的内容");

System.out.println("编码方式：" + osw.getEncoding());

osw.close();

}catch(IOException e) {

e.printStackTrace();

}

}

}

               7.输出重定向

import java.io.\*;

/\*

\* Print流属于输出流，提供了重载的方法很多，

\* PrintWriter和PrintStream不会抛异常，用户通过检测错误状态获取信息，

\* 包含自动flush功能，有什么用呢,在jsp里也要输出一些东西，

\* 但不必每次抛异常。

\*/

public class TestPrintStream {

public static void main(String[] args) {

PrintStream ps = null;

try {

FileOutputStream fos = new FileOutputStream("d:/java.txt");

ps = new PrintStream(fos);

}catch (IOException e) {

e.printStackTrace();

}

if(ps!=null) {

System.setOut(ps);//输出重定向

}

int ln = 0;

for(char c=0; c<65535; c++) {

System.out.print(c + " ");

if(ln++>100) {

System.out.println();

ln = 0;

}

}

}

}

              8.DataStream

import java.io.\*;

public class TestDataStream {

public static void main(String[] args) {

//先在内存里分配一个字节数组，再有一个 OutputStream，再加上一个数据流

ByteArrayOutputStream baos = new ByteArrayOutputStream();

DataOutputStream dos = new DataOutputStream(baos);

try {//写出读入

dos.writeDouble(Math.random());

dos.writeBoolean(true);

ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());

System.out.println(bais.available());//共几个字节可用

DataInputStream dis = new DataInputStream(bais);

////先写的先读（队列）,下面这两个输出不可以调换,否则就先输出了double里的一个字节

System.out.println(dis.readDouble());

System.out.println(dis.readBoolean());

dos.close();

dis.close();

}catch (IOException e) {

e.printStackTrace();

}

}

}

四.小问题

         为什么Writer/Reader不继承自Stream呢?字符最终也要转换成二进制呀。Writer/Readre继承OutputStream /InputStream，这样的继承层次不是更好，为什么要单独做一个呢,而且Stream也有些子类能够实现字符串的读写。大神回答：单一职责。太牵强了。

三个字

说出来很简单，也很难

我静静地坐在你身旁就是一种表达

不悲伤 不寂寞