链滴

Java IO 非详尽手册

作者：pattyq

原文链接：https://ld246.com/article/1532921656383

来源网站：链滴

JAVA IO小手册

InputStream

```
InputStream inputstream = new FileInputStream("c:\\test.txt");

int data = inputstream.read();
while(data != -1) {
  //do something with data...
  doSomethingWithData(data);

  data = inputstream.read();
}
inputstream.close();
```

try-with-resource

```
try( InputStream inputstream = new FileInputStream("test.txt") ) {

    int data = inputstream.read();
    while(data != -1){
        System.out.print((char) data);
        data = inputstream.read();
    }
}
```

读取单个字节

```
int data = inputstream.read();
//转换成char
char aChar = (char) data;
// 返回-1说明流读完了
while(data!=-1){
    //继续读
}
```

读进字节数组

```
InputStream inputstream = new FileInputStream("c:\\test.txt");

byte[] data      = new byte[1024];
int    bytesRead = inputstream.read(data);

while(bytesRead != -1) {
  doSomethingWithData(data, bytesRead);

  bytesRead = inputstream.read(data);
}
inputstream.close();
```

mark()和reset()

子类不必必须实现该方法,如果重写了该方法,那么markSupported()需要返回true,否则返回false.

```
mark(int readlimit)
        在此输入流中标记当前的位置。
reset()
        将此流重新定位到最后一次对此输入流调用 mark 方法时的位置。
```

OutputStream

写单个字节

```
OutputStream output = new FileOutputStream("c:\\test.txt");

while(hasMoreData()) {
  int data = getMoreData();
  output.write(data);
}
output.close();
```

写字节数组和flush()

```
write(byte[] b)
        将 b.length 个字节从指定的 byte 数组写入此输出流。
flush()
        刷新此输出流并强制写出所有缓冲的输出字节。
```

关闭字节流

```
OutputStream output = null;

try{
  output = new FileOutputStream("c:\\test.txt");

  while(hasMoreData()) {
    int data = getMoreData();
    output.write(data);
  }
} finally {
    try{
        if(output != null) {
            output.close();
        }
    }catch{
        // 不用操作,这是防止finally块中的output.close()尝试关闭已经关闭的流
    }
}
```

try-with-resource

```
try(output = new FileOutputStream("c:\\test.txt");){
  while(hasMoreData()) {
    int data = getMoreData();
    output.write(data);
}
```

FileInputStream

单个字节

```
InputStream input = new FileInputStream("c:test.txt");

int data = input.read();
while(data != -1) {
  doSomething(data);
  data = input.read();
}
input.close();
```

构造器

```
String path = "C:\\test.txt";

FileInputStream fileInputStream = new FileInputStream(path)

File   file = new File(path);

FileInputStream fileInputStream = new FileInputStream(file);
```

其他

```
read(byte[] b)
        从此输入流中将最多 b.length 个字节的数据读入一个 byte 数组中。
close()
        关闭此文件输入流并释放与此流有关的所有系统资源。
```

FileOutputStream

```
OutputStream output = new FileOutputStream("c:\\test.txt");

while(moreData) {
  int data = getMoreData();
  output.write(data);
}
output.close();
```

FileOutputStream构造

```
String path = "C:\\test.txt";

FileOutputStream output = new FileOutputStream(path);

// 两种
String path = "C:\\test.txt";
File   file = new File(path);

FileOutputStream output = new FileOutputStream(file);
```

追加和覆盖

```
OutputStream output = new FileOutputStream("c:\\test.txt", true); //向文件中追加
```

```
OutputStream output = new FileOutputStream("c:\\test.txt", false); //覆盖file
```

其他方法

```
write(byte[] b)
        将 b.length 个字节从指定 byte 数组写入此文件输出流中。
close()
        关闭此文件输出流并释放与此流有关的所有系统资源。
flush()  刷字节
```

RandomAccessFile

创建

rw表示读写模式(read/write)

```
RandomAccessFile file = new RandomAccessFile（"c：\\test.txt"，"rw"）；
```

从特定位置读取或写入.

```
seek(long pos)
        设置到此文件开头测量到的文件指针偏移量，在该位置发生下一个读取或写入操作。
```

```
RandomAccessFile file = new RandomAccessFile（"c：\\ test.txt"，"rw"）；

file.seek（200）；

long pointer = file.getFilePointer（）；

file.close（）；
```

读

```
RandomAccessFile file = new RandomAccessFile（"c：\\test.txt"，"rw"）；

int aByte = file.read（）；

file.close（）；
```

写

```
RandomAccessFile file = new RandomAccessFile（"c：\\test.txt"，"rw"）；

file.write（"Hello World".getBytes（））；

file.close（）；
```

ByteArrayInputStream

```
InputStream input = new ByteArrayInputStream("Hello".getBytes());

int data = input.read();
while(data != -1) {
  //do something with data
```

```
  data = input.read();
}
input.close();
```

ByteArrayOutputStream

```
ByteArrayOutputStream output = new ByteArrayOutputStream();

//write data to output stream

byte[] bytes = output.toByteArray();
```

BufferredInputStream

构造

```
InputStream input = new BufferedInputStream(new FileInputStream("c:\\test.txt"));

int bufferSize = 8 * 1024;

InputStream input = new BufferedInputStream(
                    new FileInputStream("c:\\test.txt"),
                    bufferSize
    );
```

BufferredOutputStream

构造

```
OutputStream input = new BufferedOutputStream(new FileInputStream("c:\\test.txt"));

int bufferSize = 8 * 1024;

OutputStream input = new BufferedOutputStream(
                    new FileOutputStream("c:\\test.txt"),
                    bufferSize
    );
```

SequenceInputStream

包含两个流的流

```
InputStream input1 = new FileInputStream("c:\\demo1.txt");
InputStream input2 = new FileInputStream("c:\\demo2.txt");

SequenceInputStream sequenceInputStream =
    new SequenceInputStream(input1, input2);

int data = sequenceInputStream.read();
while(data != -1){
    System.out.println(data);
```

```
    data = sequenceInputStream.read();
}
```

多个流

```
InputStream input1 = new FileInputStream("c:\\data1.txt");
InputStream input2 = new FileInputStream("c:\\data2.txt");
InputStream input3 = new FileInputStream("c:\\data3.txt");

Vector<InputStream> streams = new Vector<>();
streams.add(input1);
streams.add(input2);
streams.add(input3);

SequenceInputStream sequenceInputStream =
    new SequenceInputStream(streams.elements()))

int data = sequenceInputStream.read();
while(data != -1){
    System.out.println(data);
    data = sequenceInputStream.read();
}
sequenceInputStream.close();
```

多个流的多个流

```
SequenceInputStream sequenceInputStream1 =
        new SequenceInputStream(input1, input2);

SequenceInputStream sequenceInputStream2 =
        new SequenceInputStream(input3, input4);

SequenceInputStream sequenceInputStream =
    new SequenceInputStream(
            sequenceInputStream1, sequenceInputStream2)){

int data = sequenceInputStream.read();
while(data != -1){
    System.out.println(data);
    data = sequenceInputStream.read();
}
sequenceInputStream.close();
```

DataInputStream和DataOutputStream

读取不同类型数据

```
DataOutputStream dataOutputStream = new DataOutputStream(
                        new FileOutputStream("binary.data"));

dataOutputStream.write(45);
dataOutputStream.writeInt(4545);
dataOutputStream.writeDouble(109.123);
```

```
dataOutputStream.close();


DataInputStream dataInputStream = new DataInputStream(
                        new FileInputStream("binary.data"));

int    aByte   = input.read();
int    anInt   = input.readInt();
float  aFloat  = input.readFloat();
double aDouble = input.readDouble();
input.close();
```

DataOutputStream写,DataInputStream读

```
public class DataInputStreamExample {

    public static void main(String[] args) throws IOException {
        DataOutputStream dataOutputStream =
                new DataOutputStream(
                        new FileOutputStream("data.bin"));

        dataOutputStream.writeInt(123);
        dataOutputStream.writeFloat(123.45F);
        dataOutputStream.writeLong(789);

        dataOutputStream.close();

        DataInputStream dataInputStream =
                new DataInputStream(
                        new FileInputStream("data/data.bin"));

        int   int123     = dataInputStream.readInt();
        float float12345 = dataInputStream.readFloat();
        long  long789    = dataInputStream.readLong();

        dataInputStream.close();

        System.out.println("int123     = " + int123);
        System.out.println("float12345 = " + float12345);
        System.out.println("long789    = " + long789);
    }
}
```

ObjectInputStream和ObjectOutputStream

从文件中读取对象

```
ObjectOutputStream objectOutputStream =
    new ObjectOutputStream(new FileOutputStream("object.data"));

MyClass object = new MyClass();

output.writeObject(object);
```

```
output.close();

ObjectInputStream objectInputStream =
    new ObjectInputStream(new FileInputStream("object.data"));

MyClass object = (MyClass) objectInputStream.readObject();

objectInputStream.close();
```

ObjectInputStream读,ObjectOutputStream写

```
public class ObjectInputStreamExample {

    public static class Person implements Serializable {
        public String name = null;
        public int    age  =  0;
    }


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

        ObjectOutputStream objectOutputStream =
            new ObjectOutputStream(new FileOutputStream("data/person.bin"));

        Person person = new Person();
        person.name = "Jakob Jenkov";
        person.age  = 40;

        objectOutputStream.writeObject(person);
        objectOutputStream.close();


        ObjectInputStream objectInputStream =
            new ObjectInputStream(new FileInputStream("data/person.bin"));

        Person personRead = (Person) objectInputStream.readObject();

        objectInputStream.close();

        System.out.println(personRead.name);
        System.out.println(personRead.age);
    }
}
```

InputStreamReader

```
InputStream inputStream       = new FileInputStream("c:\\test.txt");
Reader      inputStreamReader = new InputStreamReader(inputStream);

int data = inputStreamReader.read();
while(data != -1){
    char theChar = (char) data;
    data = inputStreamReader.read();
}
```

```
inputStreamReader.close();
```

设置编码

```
InputStream inputStream       = new FileInputStream("c:\\test.txt");
Reader      inputStreamReader = new InputStreamReader(inputStream,"UTF-8");
```

OutputStreamWriter

```
OutputStream outputStream       = new FileOutputStream("c:\\test.txt");
Writer       outputStreamWriter = new OutputStreamWriter(outputStream);

outputStreamWriter.write("Hello World");

outputStreamWriter.close();
```

设置编码

```
OutputStream outputStream       = new FileOutputStream("c:\\test.txt");
Writer       outputStreamWriter = new OutputStreamWriter(outputStream,"UTF-8");
```

FileReader

```
Reader fileReader = new FileReader("c:\\test.txt");

int data = fileReader.read();
while(data != -1) {
  //do something with data...
  doSomethingWithData(data);

  data = fileReader.read();
}
fileReader.close();
```

FileWriter

```
Writer fileWriter = new FileWriter("c:\\test.txt");

fileWriter.write("data 1");
fileWriter.write("data 2");
fileWriter.write("data 3");

fileWriter.close();

// 追加或覆盖
Writer fileWriter = new FileWriter("c:\\test.txt", true);  //追加

Writer fileWriter = new FileWriter("c:\\test.txt", false); //覆盖
```

CharArrayReader

```
char[] chars = "123".toCharArray();

CharArrayReader charArrayReader =
    new CharArrayReader(chars);

int data = charArrayReader.read();
while(data != -1) {
  //do something with data

  data = charArrayReader.read();
}

charArrayReader.close();
```

CharArrayWriter

```
CharArrayWriter charArrayWriter = new CharArrayWriter();

charArrayWriter.write("CharArrayWriter");

char[] chars1 = charArrayWriter.toCharArray();

charArrayWriter.close();
```

BufferedReader

```
BufferedReader bufferedReader = new BufferedReader(
                    new FileReader("c:\\test.txt"));
int bufferSize = 8 * 1024;

BufferedReader bufferedReader = new BufferedReader(
                    new FileReader("c:\\test.txt"),
                    bufferSize
);
// 用于读取一行
String line = bufferedReader.readLine();
```

BufferedWriter

```
BufferedWriter bufferedWriter =
    new BufferedWriter(new FileWriter("c:\\test.txt"));
int bufferSize = 8 * 1024;

BufferedWriter bufferedWriter =
    new BufferedWriter(
        new FileWriter("c:\\test.txt"),
            bufferSize);
```

LineNumberReader

这个可以获取行号

行编号从 0 开始。该行号随数据读取在每个行结束符处递增，并且可以通过调用 setLineNumber(int) 更改行号。但要注意的是，setLineNumber(int) 不会实际更改流中的当前位置；它只更改将由 getLineNumber() 返回的值。

getLineNumber()
        获得当前行号。
setLineNumber(int lineNumber)
        设置当前行号。

```
LineNumberReader lineNumberReader =
   new LineNumberReader(new FileReader("c:\\data\\input.txt"));

int data = lineNumberReader.read();
while(data != -1){
   char dataChar = (char) data;
   data = lineNumberReader.read();
   int lineNumber = lineNumberReader.getLineNumber();
}
lineNumberReader.close();
```

StreamTokenizer

比如"this is an orange"作为流的source,那么StreamTokenizer的读法就是一次读一个单词.

```
StreamTokenizer streamTokenizer = new StreamTokenizer(
            new StringReader("this is an orange"));

while(streamTokenizer.nextToken() != StreamTokenizer.TT_EOF){

   if(streamTokenizer.ttype == StreamTokenizer.TT_WORD) {
      System.out.println(streamTokenizer.sval);
   } else if(streamTokenizer.ttype == StreamTokenizer.TT_NUMBER) {
      System.out.println(streamTokenizer.nval);
   } else if(streamTokenizer.ttype == StreamTokenizer.TT_EOL) {
      System.out.println();
   }
}
```

方法

```
 void commentChar(int ch)
        指定该字符参数启动一个单行注释。
 void eolIsSignificant(boolean flag)
        确定是否将行末尾视为标记。
 int lineno()
        返回当前行号。
 void lowerCaseMode(boolean fl)
        确定是否对文字标记自动使用小写字母。
 int nextToken()
        从此标记生成器的输入流中解析下一个标记。
 void ordinaryChar(int ch)
        指定字符参数在此标记生成器中是"普通"字符。
 void ordinaryChars(int low, int hi)
        指定 low <= c <= high 范围中的所有字符 c 在此标记生成器中都是"普通"字符。
```

void parseNumbers()
指定此标记生成器应解析的数字。
void pushBack()
导致对此标记生成器的 nextToken 方法的下一个调用返回 ttype 字段中的当前值，而不修改 nval 或 sval 字段中的值。
void quoteChar(int ch)
指定此字符的匹配对分隔此标记生成器中的字符串常量。
void resetSyntax()
重置此标记生成器的语法表，使所有字符都成为"普通"字符。
void slashSlashComments(boolean flag)
确定标记生成器是否识别 C++ 样式注释。
void slashStarComments(boolean flag)
确定标记生成器是否识别 C 样式注释。
String toString()
返回当前流标记和在其上发生的行号的字符串表示形式。
void whitespaceChars(int low, int hi)
指定 low <= c <= high 范围中的所有字符 c 都是空白字符。
void wordChars(int low, int hi)
指定 low <= c <= high 范围中的所有字符 c 都是文字成分。

StringReader

```
String input = "Input String... ";
StringReader stringReader = new StringReader(input);

int data = stringReader.read();
while(data != -1) {
  doSomethingWithData(data);
  data = stringReader.read();
}
stringReader.close();
```

StringWriter

```
StringWriter stringWriter = new StringWriter();

stringWriter.write("This is a text");

String       data       = stringWriter.toString();
StringBuffer dataBuffer = stringWriter.getBuffer();

stringWriter.close();// 这个关闭没什么用,关了还可以被调用
```

构造,说实话我不明白这个Writer有什么用,应该是托关系进的jdk

```
public StringWriter() {
        buf = new StringBuffer();
        lock = buf;
    }
public StringWriter(int initialSize) {
        if (initialSize < 0) {
            throw new IllegalArgumentException("Negative buffer size");
```

```
    }
    buf = new StringBuffer(initialSize);
    lock = buf;
}
```