第11讲 输 入 与 输 出

之文件的输入/输出

12.2 文件的输入/输出

12.2.1 File类

Java语言的java.io包中的File类是专门用来管理磁盘文件和目录的。每个File类的对象表示一个磁盘文件或目录，其对象属性中包含了文件或目录的相关信息，如文件或目录的名称、文件的长度、目录中所含文件的个数等。调用File类的方法则可以完成对文件或目录的常用管理操作，如创建文件或目录、删除文件或目录、查看文件的有关信息等。

java.io.File 类的父类是java.lang.Object。用于创建File类对象的构造方法有三个，它们分别是：

public File(String path); //使用指定路径构造一个File对象

public File(String path,String name); //使用指定路径和字符串构造一个File对象

public File(File dir,String name,); //使用指定文件目录和字符串构造一个File对象

表14.7 File类成员方法

成 员 方 法	说 明
public boolean canRead( )	测试应用程序是否能从指定的文件读
public boolean canWrite( )	测试应用程序是否能写指定的文件
public boolean delete( )	删除此对象指定的文件
public boolean exists( )	测试文件是否存在
public String getAbsolutePath( )	获取此对象表示的文件的绝对路径名
public String getCanonicalPath( ) Throws IOException	获取此文件对象路径名的标准格式
public String getName( )	获取此对象代表的文件名
public String getParent( )	获取此文件对象的路径的父类部分
public String getPath( )	获取此对象代表的文件的路径名
public native boolean isAbsolute( )	测试此文件对象代表的文件是否是绝对路径

续表

成 员 方 法	说　　明
public boolean isDirectory( )	测试此文件对象代表的文件是否是一个目录
public boolean isFile( )	测试此文件对象代表的文件是否是一个“正常”文件
public long length( )	获取此文件对象代表的文件长度
public String[ ] list(Filename filter)	获取在文件指定的目录中满足指定过滤器的文件列表
public String[ ] list( )	获取在此文件对象指定的目录中的文件列表
public boolean mkdir( )	创建一个目录，其路径名由此文件对象指定
public boolean mkdirs( )	创建一个目录，其路径名由此文件对象指定并包括必要的父目录
public boolean renameTo(File dest)	将此文件对象指定的文件换名为由文件参数给定的文件名
public String toString( )	获取此对象的字符串表示

【示例程序c12_5.java】 获取文件的文件名、长度、大小等特性。

```
import java.io.*;
import java.util.*;
public class c12_5
{   public static void main(String args[ ])
  {
     String Path;
     InputStreamReader din=new InputStreamReader(System.in);  //键盘输入
     BufferedReader in=new BufferedReader(din);
     try{
```


```
System.out.print("请输入相对或绝对路径:  ");
Path=in.readLine( );     //读取输入
File f = new File(Path);
System.out.println("路径: "+f.getParent( ));
System.out.println("档案: "+f.getName( ));
System.out.println("绝对路径: "+f.getAbsolutePath( ));
System.out.println("文件大小: "+f.length( ));
System.out.println("是否为文件: "+(f.isFile( )?"是":"否"));
System.out.println("是否为目录: "+(f.isDirectory( )?"是":"否"));
```

```
            System.out.println("是否为隐藏: "+(f.isHidden( )?"是":"否"));
            System.out.println("是否可读取: "+(f.canRead( )?"是":"否"));
            System.out.println("是否可写入: "+(f.canWrite( )?"是":"否"));
            System.out.println("最后修改时间: "+new Date(f.lastModified( )));
        }
        catch(IOException E)
        { System.out.println("I/O错误!");  }
    }
}
```

该程序的运行结果如下：

请输入相对或绝对路径: 路径: null

档案: c12_4.java

绝对路径: D:\ZBYJAVA\javabook\ch14\c14_4.java

文件大小: 1234

是否为文件: 是

是否为目录: 否

是否为隐藏: 否

是否可读取: 是

是否可写入: 是

最后修改时间: Thu Aug 11 17:06:40 CST 2005

12.2.2 FileInputStream类和FileOutputStream类

在程序中，经常会用到文件的读/写操作。例如，从已经存在的数据文件中读入数据，或者将程序中产生的大量数据写入磁盘文件中。这时就需要使用文件输入/输出流类。Java系统提供的FileInputStream类是用于读取文件中的字节数据的字节文件输入流类；FileOutputStream类是用于向文件写入字节数据的字节文件输出流。表14.8列出了FileInputStream类和FileOutputStream类的构造方法，表14.9列出了这两个类的常用成员方法。

表12.8 FileInputStream类和FileOutputStream类的构造方法

构造方法	说明
public FileInputStream(String name)	使用指定的字符串创建一个 FileInputStream 对象
public FileInputStream(File file)	使用指定的文件对象创建一个 FileInputStream 对象
public FileInputStream(FileDescriptor fdObj)	使用指定的 FileDescriptor 创建一个 FileInputStream 对象
public FileOutputStream(String name)	使用指定的字符串创建 FileOutputStream 对象
public FileOutputStream(File file)	使用指定的文件对象创建 FileOutputStream 对象
public FileOutputStream(FileDescriptor fdObj)	使用指定的 FileDescriptor 创建 FileOutputStream 对象

表12.9 FileInputStream类和FileOutputStream类的常用成员方法

成员方法	说明
public int read( )throws IOException	自输入流中读取一个字节
public int read(byte b[ ])throws IOException	将输入数据存放在指定的字节数组 b 中
public int read(byte b[ ],int offset,int len) throws IOException	自输入流中读取 len 个字节并存放在指定的数组 b 的 offset 开始位置中
public int available( )throws IOException	返回输入流中的可用字节个数
public long skip(long n) throws IOException	从输入流中跳过 n 个字节
public void write(int b) throws IOException	写一个字节
public void write (byte b[ ])throws IOException	写一个字节数组
public void write (byte b[ ],int offset,int len) throws IOException	将字节数组 b 从 offset 位置开始的 len 个字节数组的数据写到输出流中
public void close( )throws IOException	关闭输入/输出流，释放占用的所有资源
public final FileDescriptor getFD( )throws IOException	获取与此流关联的文件描述符

14.2.3 字节文件输入/输出流的读写

1. 用文件输入/输出类自身的读写功能完成文件的读写操作

FileInputStream类和FileOutputStream类自身的读写功能是直接从父类InputStream和OutputStream那里继承来的，并未做任何功能的扩充。如前面介绍过的read( )，write( )等方法，都只能完成以字节为单位的原始二进制数据的读写。

2．配合其他功能较强的输入/输出流完成文件的读写操作

以FileInputStream和FileOutputStream为数据源，完成与磁盘文件的映射连接后，再创建其他流类的对象，如DataInputStream类和DataOutputStream类，这样就可以从FileInputStream和FileOutputStream对象中读写数据。利用DataInputStream类和DataOutputStream类提供的成员方法(见表14.2和表14.4)可以方便地从文件中读写不同类型的数据。其使用方式如下：

```
File f＝new File("TextFile");
DataInputStream din＝new DataInputStream(new FileInputStream(f));
DataOutputStream dout＝new DataOutputStream(new FileOutputStream(f));
```

【示例程序c12_7.java】 直接利用FileInputStream类和FileOutputStream类完成从键盘读入数据写入文件中，再从写入的文件中读出数据打印到屏幕上的操作。

```
import java.io.*;
public class c12_7
{
  public static void main(String [ ] args)
    {
      char c;
       int c1;
      File filePath=new File("temp");  //在当前目录下建目录，也可用绝对目录
```

```
if(!filePath.exists( ))filePath.mkdir( );  //若目录不存在，则建之
File fl=new File(filePath,"d1.txt");  //在指定目录下建文件类对象
try {
   FileOutputStream fout=new FileOutputStream(fl);
   System.out.println("请输入字符,输入结束按# :");
   while((c=(char)System.in.read( ))!='#')  //键盘输入字符写到磁盘文件
        {  fout.write(c);  }
   fout.close( );  //
   System.out.println("\n打印从磁盘读入的数据");
```

```
FileInputStream fin=new FileInputStream(fl);
 while((c1=fin.read( ))!=-1)   //磁盘文件读入程序
        System.out.print((char)c1);  //打印从磁盘读入的数据到屏幕上
 fin.close( );
} //try结束
catch(FileNotFoundException e) {
   System.err.println(e);}
catch(IOException e){
   System.err.println(e);}
} //main( )结束
} //class c12_7结束
```

运行结果如下:

请输入字符,输入结束按#:

12345

abcd#

打印从磁盘读入的数据

12345

abcd

【示例程序c12_8.java】 用FileInputStream和FileOutputStream输入输出流，再套接上DataInputStream类和DataOutputStream类输入输出流完成文件的读写操作。

```
import java.io.*;
public class c12_8
{
  public static void main(String args[ ])
  {
    boolean lo=true;  short si=-32768;
```

```
int i=65534;     long l=144567;

float f=(float)1.4567;   double d=3.14159265359;

String str1="ABCD";      String str2="Java语言教学";

try
{   FileOutputStream fout=new FileOutputStream("t1.txt");
    DataOutputStream out=new DataOutputStream(fout);  //文件输出流对象为参数
    FileInputStream fin=new FileInputStream("t1.txt");
    DataInputStream in=new DataInputStream(fin);  //文件输入流对象为参数
```

```
out.writeBoolean(lo); out.writeShort(si);
out.writeByte(i);     out.writeInt(i);
out.writeLong(l);     out.writeFloat(f);
out.writeDouble(d);   out.writeBytes(str1);
out.writeUTF(str2);
out.close( );
System.out.println("Blooean lo="+in.readBoolean( ));
System.out.println("Short si="+in.readShort( ));
System.out.println("Byte i="+in.readByte( ));
System.out.println("Int i="+in.readInt( ));
System.out.println("Long l="+in.readLong( ));
System.out.println("Float f="+in.readFloat( ));
System.out.println("Double d="+in.readDouble( ));
```

```
            byte b[ ]=new byte[4];
            in.readFully(b);
            System.out.print("str1=");
            for(int j=0;j<4;j++)System.out.print((char)b[j]);
            System.out.println( );
            System.out.println("str2="+in.readUTF( ));
            in.close( );
        }
        catch(IOException E)
        {
            System.out.println(E.toString( ));
        }
    }
}
```

运行结果如下:

Boolean lo=true　Short si=-32768

Byte i=-2　　Int i=65534

Long l=144567　　Float f=(float)1.4567

Double d=3.14159265359

str1=ABCD　　str2=Java语言教学

12.2.4 FileReader类和FileWriter类

FileReader类和FileWriter类用于读取文件和向文件写入字符数据。表12.10列出了FileReader类和FileWriter类的构造方法。

表12.10 FileReader类和FileWriter类的构造方法

构 造 方 法	说　　明
publicFileReader (String fileName)throws FileNotFoundException	使用指定的文件名创建一个 FileReader 对象
Public FileReader (File file)throws FileNotFoundException	使用指定的文件对象创建一个 FileReader 对象
publicFileReader (FileDescriptor fd)	使用指定的文件描述符创建一个 FileReader 对象
PublicFileWriter (String fileName)throws IOException	使用指定的文件名创建一个 FileWriter 对象
PublicFileWriter (File file) throws IOException	使用指定的文件对象创建一个 FileWriter 对象
PublicFileWriter (FileDescriptor fd)	使用指定的文件描述符创建一个 FileWriter 对象

【示例程序c12_9.java】 复制文件。

```
import java.io.*;
public class c12_9
{
   public static void main(String args[ ])
   {
      String temp;
      File   sourceFile,targetFile;  //创建File对象
      BufferedReader source;
      BufferedWriter target;
      try
      {
```

```
InputStreamReader din = new InputStreamReader(System.in);
BufferedReader in = new BufferedReader(din);
System.out.print("请输入来源文件路径: ");
sourceFile = new File(in.readLine( ));
source = new BufferedReader(new FileReader(sourceFile)); //
System.out.print("请输入目的文件路径: ");
targetFile = new File(in.readLine( ));
target = new BufferedWriter(new FileWriter(targetFile));
System.out.print("确定要复制?(y/n) ");
if((in.readLine( )).equals("y"))
{
   while((temp=source.readLine( )) != null)
```

```
        {  target.write(temp);
           target.newLine( );
           target.flush( );
        }
        System.out.println("复制文件完成!!!");
    }
    else
    {
        System.out.println("复制文件失败!!!");
        return;
    }
    din.close( );
    in.close( );
}
```

```
        catch(IOException E)
        {
            System.out.println("I/O错误!");
        }
    }
}
```

运行结果：

请输入来源文件路径: d:\zby\t1.txt

请输入目的文件路径: t2.txt

确定要复制?(y/n) y

复制文件完成!!!

结果是将t1.txt复制到t2.txt中。程序中使用FileReader类输入流套接BufferedReader类缓冲区输入流、FileWriter类输出流套接BufferedWriter类缓冲区输出流的策略，加快了复制文件的速度。

14.2.5 RandomAccessFile类

前面介绍的文件存取方式属于顺序存取，即只能从文件的起始位置向后顺序读写。java.io包提供的RandomAccessFile类是随机文件访问类, 该类的对象可以引用与文件位置指针有关的成员方法，读写任意位置的数据，实现对文件的随机读写操作。文件的随机存取要比顺序存取更加灵活。

1. java.io.RandomAccessFile类的构造方法

java.io.RandomAccessFile类的构造方法有两种：

(1) RandomAccessFile(String name,String mode) 使用指定的字符串和模式参数创建一个RandomAccessFile类对象。

(2) RandomAccessFile(File f,String mode) 使用指定的文件对象和模式参数创建一个RandomAccessFile类对象。

在RandomAccessFile类的构造方法中，除了指定文件的路径外，还必须指定文件的存取模式。存取模式有读模式和读写模式两种：“r”代表以只读方式打开文件；“rw”代表以读写方式打开文件，这时用一个对象就可以同时实现读写两种操作。需要注意的是，创建RandomAccessFile对象时，可能产生两种异常：当指定的文件不存在时，系统将抛出FileNotFoundException异常；若试图用读写方式打开具有只读属性的文件或出现了其他输入/输出错误时，则会抛出IOException异常。

2. RandomAccessFile类中的常用成员方法

表12.11 RandomAccessFile类中的常用成员方法

成员方法	说明
Public native long getFilePointer( ) throws IOException	取得文件的指针
Public native long length( ) throws IOException	以字节为单位获取文件的大小
Public native int read( )throws IOException	自输入流中读取一个字节
Public int read(byte b[ ])throws IOException	将输入的数据存放在指定的字节数组 b[ ]中
Public int read(byte b[ ],int offset,int len) throws IOException	自输入流的 offset 位置开始读取 len 个字节并存放在指定的数组 b[ ]中
Public void write(int b) throws IOException	写一个字节
Public void write (byte b[ ])throws IOException	写一个字节数组
Public void write (byte b[ ],int offset,int len) throws IOException	将字节数组 b[ ]中从 offset 位置开始的长度为 len 个字节的数据写到输出流中
Public native void close( )throws IOException	关闭数据流
Public native void seek(long pos)throws IOException	将文件位置指针置于 pos 处，pos 以字节为单位

3．对文件位置指针的操作

RandomAccessFile类的对象可以引用与文件位置指针有关的各种成员方法，在任意位置实现数据读写。RandomAccessFile类对象的文件位置指针遵循以下规律：

(1) 新建RandomAccessFile对象文件位置指针位于文件的开头处。

(2) 每次读写操作之后，文件位置指针都后移相应个读写的字节数。

(3) 利用seek( )方法可以移动文件位置指针到一个新的位置。

(4) 利用getPointer( )方法可获得本文件当前的文件位置指针。

(5) 利用length( )方法可得到文件的字节长度。利用getPointer( )方法和length( )方法可以判断读取的文件是否到文件尾部。

【示例程序c12_10.java】 从键盘输入五个整数写入文件t1.txt中，再从这个文件中随机读出其中的某个数(由键盘输入确定)，将它显示在屏幕上，同时允许用户对这个数进行修改。

```
import java.io.*;
public class c12_10
{
   public static void main(String args[ ])
   {  int  num,a;
      long fp;
      try
      {
```

```
InputStreamReader din=new InputStreamReader(System.in);  //键盘输入
BufferedReader in=new BufferedReader(din);
//建立随机存取文件
RandomAccessFile rf=new RandomAccessFile(args[0],"rw");
System.out.println("请输入五个整数");
int b[ ]=new int[5];
for(int i=0;i<5;i++)
{
   System.out.print("第" + (i+1)+"个数   ");
   b[i]=Integer.parseInt(in.readLine( ));
   rf.writeInt(b[i]);      //写入文件
}
```

```
while(true)
{  fp=0;
   rf.seek(0);     //移动文件指针到文件头
   System.out.print("请输入要显示第几个数(1-5)：");
   num=Integer.parseInt(in.readLine( )); //读入序号
   num=num-1;
   fp=(num)*4;   //每个整数4个字节，计算移动位数
   rf.seek(fp);     //移动文件指针到要显示数的首位
   a=rf.readInt( );
   System.out.println("第"+(num+1)+"个数是: "+a);
   System.out.print("改写此数");
   b[num]=Integer.parseInt(in.readLine( ));
   fp=num*4; rf.seek(fp);
   rf.writeInt(b[num]);      //写入文件
```

```
            System.out.print("继续吗?(y/n) ");
             if((in.readLine( )).equals("n")) break;
          }
          rf.close( );
        }
        catch(Exception E)
        {   System.out.println("I/O错误!");  }
      }
    }
```

运行过程如下：

```
JAVA C14_10 T1.TXT
请输入五个整数
第1个数  11
第2个数  22
第3个数  33
第4个数  44
第5个数  55
请输入要显示第几个数(1-5)：2
第2个数是: 22
改写此数  222
继续吗?(y/n) y
请输入要显示第几个数(1-5)：2
第2个数是: 222
改写此数  222
继续吗?(y/n) n
```