第10课

输入/输出

SOFTCITS@2016

SoftCITS SOFT CULTURE IT SALON 软件园IT文化沙龙

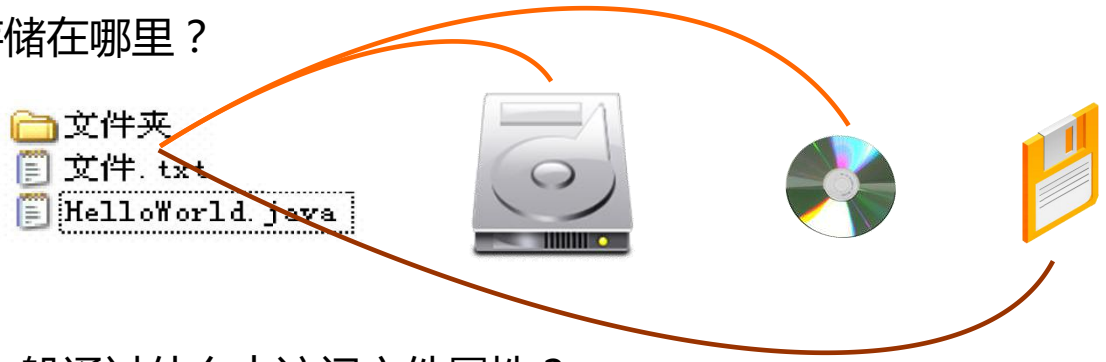
文件

1. 什么是文件？

   文件可认为是相关记录或放在一起的数据的

   集合

2. 文件一般存储在哪里？



3. JAVA程序一般通过什么去访问文件属性？
   JAVA API ：java.io.File 类

文件

1. File类可以使用文件路径字符串来创建File实例，该文件路径字符串既可以是绝对路径，也可以是相对路径，默认情况下，系统总是依据用户的工作路径来解释相对路径，这个路径由系统属性“user.dir”指定，通常也就是运行Java虚拟机时所在的路径。

```
import java.io.File;
File file = new File(“XXXXX");
// to get the object file path
System.out.println(file.getAbsoluteFile());
 //to get the current user working path
System.getProperty("user.dir")
```

2. File能够新建、删除、重命名文件和目录，不能访问文件内容本身。文件内容本身的操作，需要使用输入/输出流

文件

String getName() 返回文件、路径名

String getPath() 返回相对路径

String getAbsolutePath() 返回绝对路径

boolean exists(); 判断File对象是否存在

boolean isFile()/isDirectory(); 是否是文件/目录

long lastModified(); 返回文件的最后修改时间

boolean createNewFile()/mkdir(); 创建新文件/目录

boolean delete();删除文件或路径

String[] list(); 列出所有子文件和路径

示例代码 1.1

文件过滤器

1. 在File的list方法中可以接受一个FilenameFilter参数，通过该参数可以只列出符合条件的文件。
2.FilenameFilter接口里包含了一个accept(File dir, String name)抽象方法，该方法将依次对指定File的子目录、子文件夹进行迭代，如果该方法返回true则list方法会列出该子目录或者子文件

示例1.2 – 列出目录下所有可读文件夹和文本文件

IO流

1. Java把不同的输入、输出源（键盘、文件、网络传输等）表述为 ‘流’ (stream)，并封装在java.io包中。

2.输入流、输出流

3.字节流、字符流

字节流的操作单元是8位字节、字符流的数据单元是16位字节。

字节流的基类：InputStream/OutputStream

字符流的基类：Reader/Writer

常用的流分类

分类	字节输入流	字节输出流	字符输入流	字符输出流
抽象基类	InputStream	OutputStream	Reader	Writer
文件	FileInputStream	FileOutputStream	FileReader	FileWriter
数组	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
管道	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
字符串			StringReader	StringWriter
缓冲流	BufferedInputStream	BufferedOuputStream	BufferedReader	BufferedWriter
转换流			InputStreamReader	OutputStreamWriter
打印流		PrintSteam		PrintWriter
对象流	ObjectInputStream	ObjectOutputStream		

输入输出流

OutputStream和Writer是所有输出流的基类，两个流都提供了如下三个方法：

void write(int c)：将指定的字节/字符输出到输出流中，其中c既可以代表字节，也可以代表字符。

void write(byte[]/char[] buf)：将字节数组/字符数组中的数据输出到指定输出流中。

void write(char[] cbuf, int off, int len)：将字节数组/字符数组中从off位置开始，长度为len的字节/字符输出到输出流中。

InputStream和Reader是所有输入流的基类，它们都是两个抽象类，本身并不能创建实例来执行输入，它们包含如下三个方法：

int read()：从输入流中读取单个字节，返回所读取的字节数。

int read(byte[]/char[] b)：从输入流中读取最多b.length个字节的数据，并将其存储在字节数组b中，返回实际读取的字节数.

int read(byte[]/char[] b, int off, int len)：从输入流中读取最多len字节的数据，并将其存储在数组 b 中，放入b数组中时，并不是从数组起点开始，而是从off位置开始，返回实际读取的字节数。

示例1.3.1， 1.3.2

处理流及转换流

之前介绍的IO都是节点流，也称低级流(Low Level Stream),程序和实际的输入、输出节点连接。

处理流则用于对节点流封装，不关心具体物理节点，从而消除不同节点的差异，采用完全相同的输入、输出代码来访问不同的数据源，并以增加缓冲的方式来提高IO效率。

输入/输出流体系里还提供了2个转换流，这两个转换流用于实现将字节流转换成字符流，其中InputStreamReader将字节输入流转换成字符输入流，OutputStreamWriter将字节输出流转换成字符输出流。

示例代码-1.3.3

重定向标准输入/输出

System类里提供了三个重定向标准输入/输出的方法：

static void setErr(PrintStream err)：重定向 “标准”错误输出流。

static void setIn(InputStream in)：重新分配“标准”输入流。

static void setOut(PrintStream out)：重定向 “标准”输出流。

示例代码-1.4.1 1.4.2

RandomAccessFile

RandomAccessFile支持“随机访问”的方式，程序可以直接跳转到文件的任意地方来读写数据,可以向已存在的文件后追加内容。

一个最大的局限，只能读写文件，不能读写其他IO节点

RandomAccessFile对象也包含了一个记录指针，用以标识当前读写处的位置，当程序新创建一个RandomAccessFile对象时，该对象的文件记录指针位于文件头（也就是0处），当读/写了n个字节后，文件记录指针将会向后移动n个字节。除此之外，RandomAccessFile可以自由移动该记录指针，既可以向前移动，也可以向后移动。RandomAccessFile包含了如下两个方法来操作文件记录指针：

long getFilePointer()：返回文件记录指针的当前位置。

void seek(long pos)：将文件记录指针定位到pos位置。

示例代码-1.5.1 1.5.2

RandomAccessFile—插入文件

RandomAccessFile依然不能向文件的指定位置插入内容，如果直接将文件记录指针移动到中间某位置后开始输出，则新输出的内容会覆盖文件中原有的内容。
如果需要向指定位置插入内容，程序需要先把插入点后面内容读入缓冲区，等将需要插入的数据写入文件后，再将缓冲区的内容追加到文件后面。

示例代码 – 1.5.3

序列化、反序列化

对象的序列化（Serialize）指将一个Java对象写入IO流中，与此对应的是，对象的反序列化（Deserialize）则指从IO流中恢复该Java对象。
如果需要让某个对象可以支持序列化机制，必须让它的类是可序列化的（serializable），为了让某个类是可序列化的，该类必须实现如下两个接口之一：

Serializable – 只是个标记接口，实现该接口无需实现任何方法
Externalizable

示例代码-1.6.1

1. 反序列化时必须按序列化顺序读取
2.序列化的类成员变量若是引用类型而非基本类型，该引用类必须可序列化

示例代码- 1.6.2

序列化有多个父类时，父类必须满足其一
1.父类可序列化
2.父类有无参数构造器 – 但成员变量值不会被序列化

自定义序列化

应用在对序列对象的特殊处理，如加密解密

在序列化和反序列化过程中需要特殊处理的类应该提供如下特殊签名的方法，这些特殊的方法用以实现自定义序列化：

```
private void writeObject(java.io.ObjectOutputStream out)throws IOException
private void readObject(java.io.ObjectInputStream in)
        throws IOException, ClassNotFoundException;
```

注意writeObject和readObject对实列变量的存储和恢复顺序保持一致

示例代码 – 1.6.3

新IO(NIO)- java.nio

新IO将文件或文件的一段区域映射到内存中，这样就可以像访问内存一样来访问文件了，通过这种方式来进行输入/输出比传统的输入/输出要快得多。相比传统数据流操作IO,NIO更像是对块操作。

Channel 和 Buffer 是新IO的核心

Channel
所有数据要通过Channel传输到Buffer，然后读入程序或写入磁盘。 实现类FileChannel,FileChannel 不能通过构造器来直接创建，而需使用传统节点的getChannel方法。

Buffer
容量（capacity）：缓冲区的 容量（capacity） 表示该Buffer的最大数据容量，即最多可以存储多少数据。
界限（limit）：标记不应该被读出或者写入的缓冲区位置索引。也就是说，位于limit后的数据既不可被读，也不可被写。
位置（position）：用于指明下一个可以被读出的或者写入的缓冲区位置索引，初始为0.

flip(): 为可读作准备，将limit设为position,并将position设为0
Clear():为可写作准备，将position设为0，讲limit设为capacity

0<= position<=limit<=capacity

示例代码-1.7.1

谢谢观看

SeeYou！

SoftCITS 软件园IT文化沙龙

SOFT CULTURE IT SALON