

JAVA编程高级

—— 文件与流



Programming
Your Future



Programming Your Future

本章内容

节	知识点	掌握程度	难易程度
文件管理	文件管理概述	掌握	
	File类	掌握	
流的概念及API	流的概念	掌握	
	字节流中的层次结构图	掌握	
	字节流中的主要方法	掌握	
	字符流中的层次结构图	掌握	
	字符流中的主要方法	掌握	
节点流与处理流的使用	什么是节点流	掌握	
	节点流的方法	掌握	
	文件的访问	掌握	
	什么是处理流	掌握	
	常见的处理流类	掌握	
对象的序列化	对象序列化概述	掌握	
	支持序列化的接口和类	掌握	难
	对象序列化的条件	掌握	难
	transient	理解	

文件管理

- 文件管理的概述

- ✓ Java中的对文件的管理，通过java.io包中的File类实现
- ✓ Java中文件的管理，主要是针对文件或是目录路径名的管理
 - 文件的属性信息
 - 文件的检查
 - 文件的删除等
 - 不包括文件的访问

文件管理

- **File类**

- ✓ File类的构造方法

```
File 变量名 = new File(String pathname) ;
```

- 通过将给定路径名字符串转换成抽象路径名来创建一个新File 实例

```
File f1 = new File ("d:/temp/abc.txt");
```

文件管理

- **File类**

- ✓ File类的构造方法

```
File 变量名 = new File(URI uri);
```

- 通过将给定File的uri转换成抽象路径名来创建一个新 File 实例

```
File f2 = new File("abc.txt");
```

文件管理

- **File类**

- ✓ File类的构造方法

```
File 变量名 = new File(String parent, String child) ;
```

- 根据 parent 路径名字符串和 child 路径名字符串创建一个新 File 实例

```
File f3 = new File("d:/temp","abc.txt");
```

文件管理

- **File类**

- ✓ File类的构造方法

```
File 变量名 = new File(File parent, String child) ;
```

- 根据 parent 抽象路径名和 child 路径名字符串创建一个新 File 实例

```
File f = new File("d:/temp");  
File f4 = new File(f, "abc.txt");
```


文件管理

- **File类**

方法	含义
<code>boolean createNewFile()</code>	当且仅当不存在具有此抽象路径名指定的名称的文件时，原子地创建由此抽象路径名指定的一个新的空文件。
<code>static File createTempFile(String prefix,String suffix)</code>	在默认临时文件目录中创建一个空文件，使用给定前缀和后缀生成其名称
<code>static File createTempFile(String prefix,String suffix,File directory)</code>	在指定目录中创建一个新的空文件，使用给定的前缀和后缀字符串生成其名称

文件管理

- **File类**

方法	含义
boolean exists()	测试此抽象路径名表示的文件或目录是否存在
boolean delete()	删除此抽象路径名表示的文件或目录
boolean equals(Object obj)	测试此抽象路径名与给定对象是否相等
boolean canRead()	测试应用程序是否可以读取此抽象路径名表示的文件
boolean canWrite()	测试应用程序是否可以修改此抽象路径名表示的文件
String[] list()	返回由此抽象路径名所表示的目录中的文件和目录的名称所组成字符串数组

文件管理

- File类

示例 FileDemo.java

方法	含义
String getAbsolutePath()	返回抽象路径名的绝对路径名字符串
String getName()	返回由此抽象路径名表示的文件或目录的名称，不包括路径名称
String getPath()	将此抽象路径名转换为一个路径名字符串
File[] listFiles()	返回一个抽象路径名数组，这些路径名表示此抽象路径名所表示目录中的文件
boolean renameTo(File dest)	重新命名此抽象路径名表示的文件
long length()	返回由此抽象路径名表示的文件的大小，以byte为单位

文件管理

- **File类**

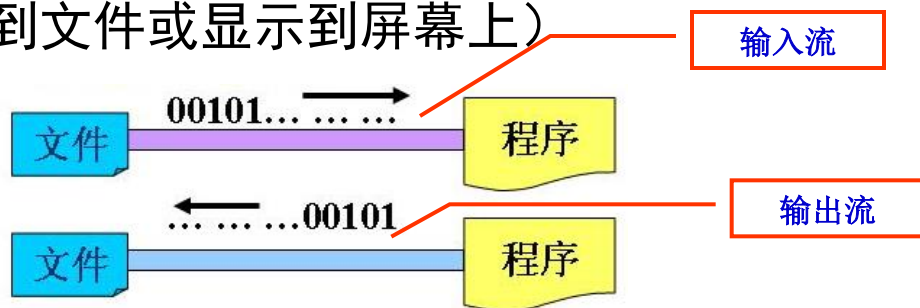
示例 FileDemo.java

方法	含义
boolean mkdir()	创建此抽象路径名指定的目录
boolean mkdirs()	创建此抽象路径名指定的目录，包括创建必需但不存在的父目录。注意，如果此操作失败，可能已成功创建了一些必需的父目录

流的概念及API

- 流的概念

- ✓ 流 (Stream) 的概念代表的是程序中数据的流通
- ✓ 数据流是一串连续不断的数据的集合
- ✓ 在Java程序中，对于数据的输入/输出操作是以流(Stream)的方式进行的
 - 输入流 — 流入程序的数据
 - 输出流 — 流出程序的数据
 - 在java程序中，从输入流读取数据（读入内存中），而从输出流输出数据（从内存存储到文件或显示到屏幕上）



流的概念及API

• 流的概念

✓ 流的分类

➤ 按流的方向不同

☞ 输入流、输出流

➤ 按处理数据的单位不同

☞ 字节流、字符流

➤ 按功能不同

☞ 节点流、处理流

- ✓ Java语言中，控制数据流的类都放在java.io包中

	字节流	字符流
输入流	InputStream	Reader
输出流	OutputStream	Writer

✓ java.io包中有两大继承体系

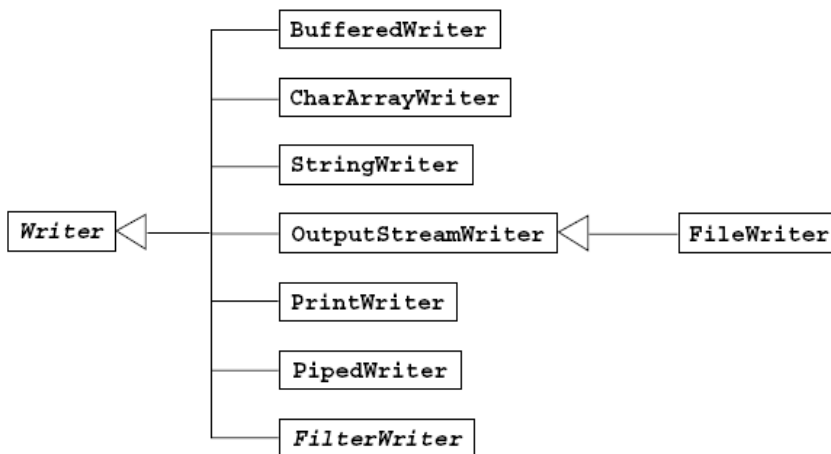
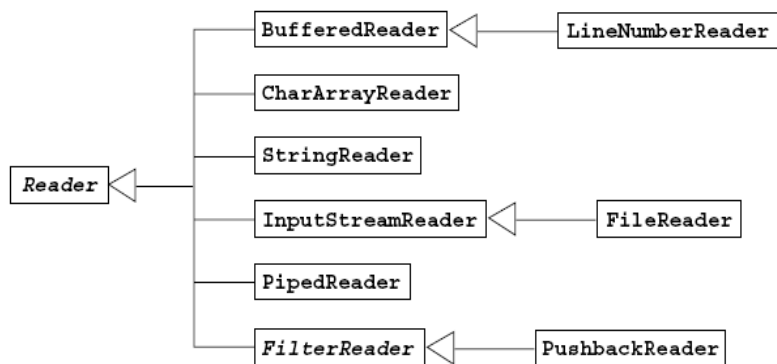
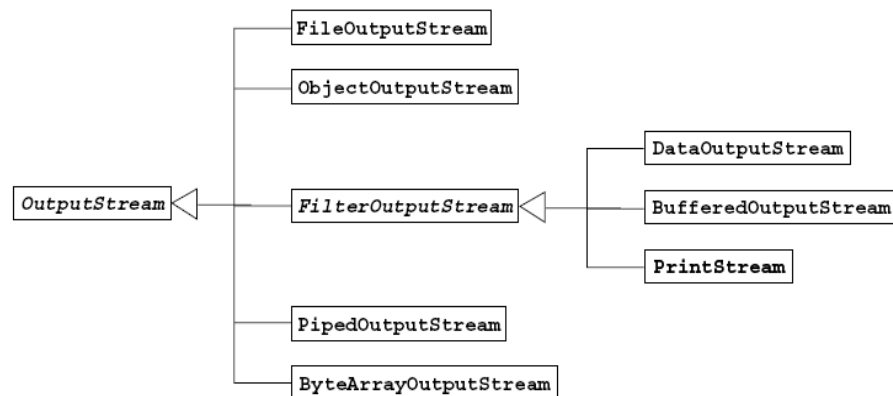
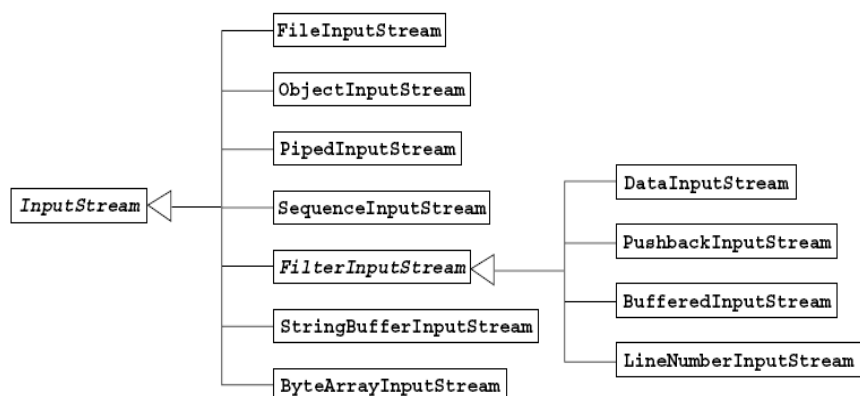
➤ 以byte处理为主的Stream类，他们的命名方式是XXXStream

➤ 以字符处理为主的Reader / Writer类，他们的命名方式XXXReader或XXXWriter

- ✓ InputStream、OutputStream、Reader、Writer这四个类，是这两大继承体系的父类

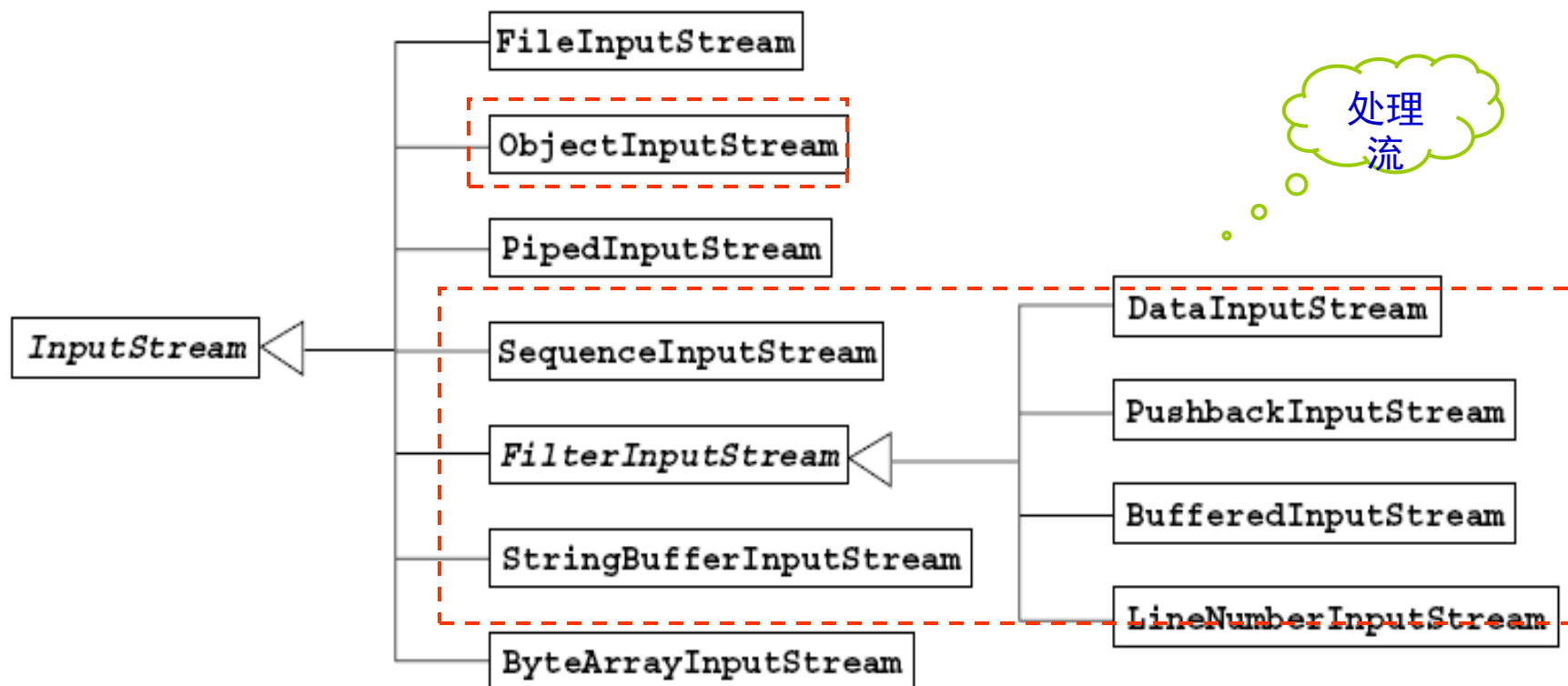
流的概念及API

- 流的层次结构图



流的概念及API

• 字节输入流的层次结构图



流的概念及API

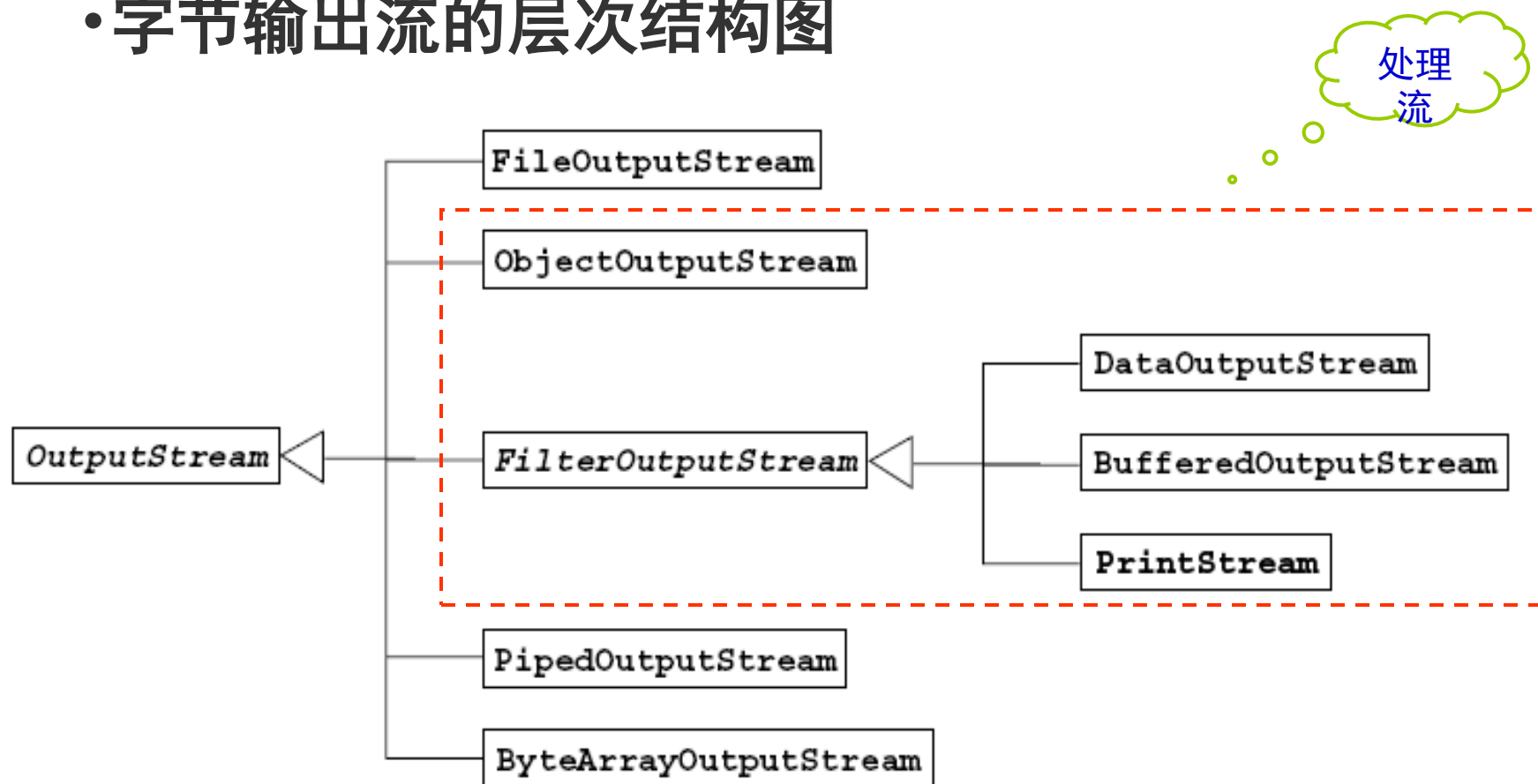
- 字节输入流的主要方法

- ✓ 此抽象类是表示**输入字节流**的所有类的超类
- ✓ InputStream常用的方法

方法	含义
<code>int read()</code>	一次读取一个byte的数据，并以int类型把数据返回来，如果没有数据可以读了，会返回” -1”
<code>int read(byte[] buffer)</code>	把所读取到的数据放在这个byte数组中，返回一个int型的数据，这个int型数据存储了返回的真正读取到的数据byte数
<code>int read(byte[] buffer,int offset,int length)</code>	读取length个字节，并存储到一个字节数组buffer中，并从offset位置开始返回实际读取的字节数
<code>void close()</code>	关闭此输入流并释放与该流关联的所有系统资源

流的概念及API

• 字节输出流的层次结构图



流的概念及API

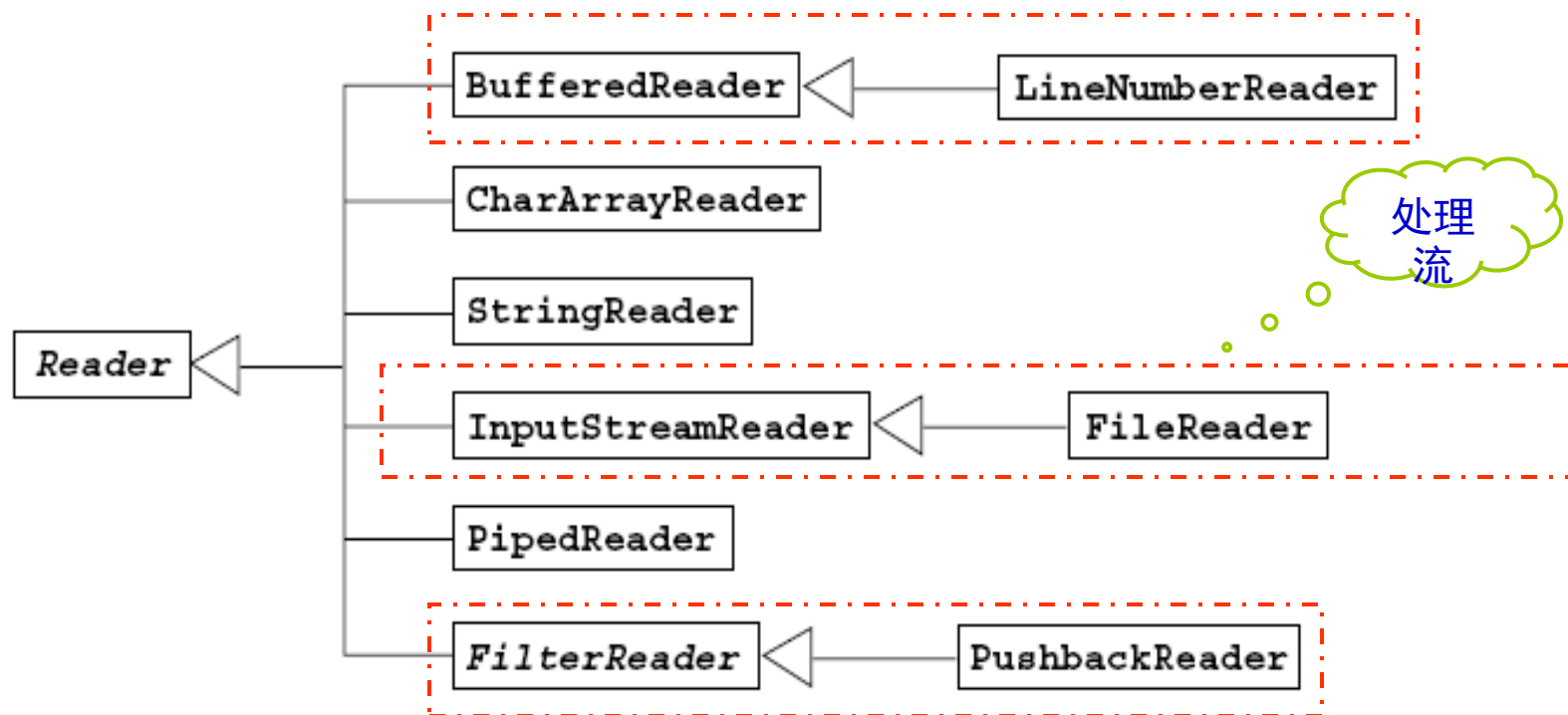
- 字节输出流的主要方法

- ✓ 此抽象类是表示输出字节流的所有类的超类
- ✓ OutputStream常用的方法

方法	含义
void write(byte[] buffer)	将要输出的数组先放在一个byte数组中，然后用这个方法一次把一组数据输出出去
void write(byte[] buffer,int off,int len)	将指定字节数组中从偏移量 off 开始的 len 个字节写入此输出流
abstract void write(int b)	将指定的字节写入此输出流
void close()	关闭此输出流并释放与此流有关的所有系统资源
void flush()	刷新此输出流并强制写出所有缓冲的输出字节

流的概念及API

• 字符输入流的层次结构图



流的概念及API

- 字符输入流的主要方法

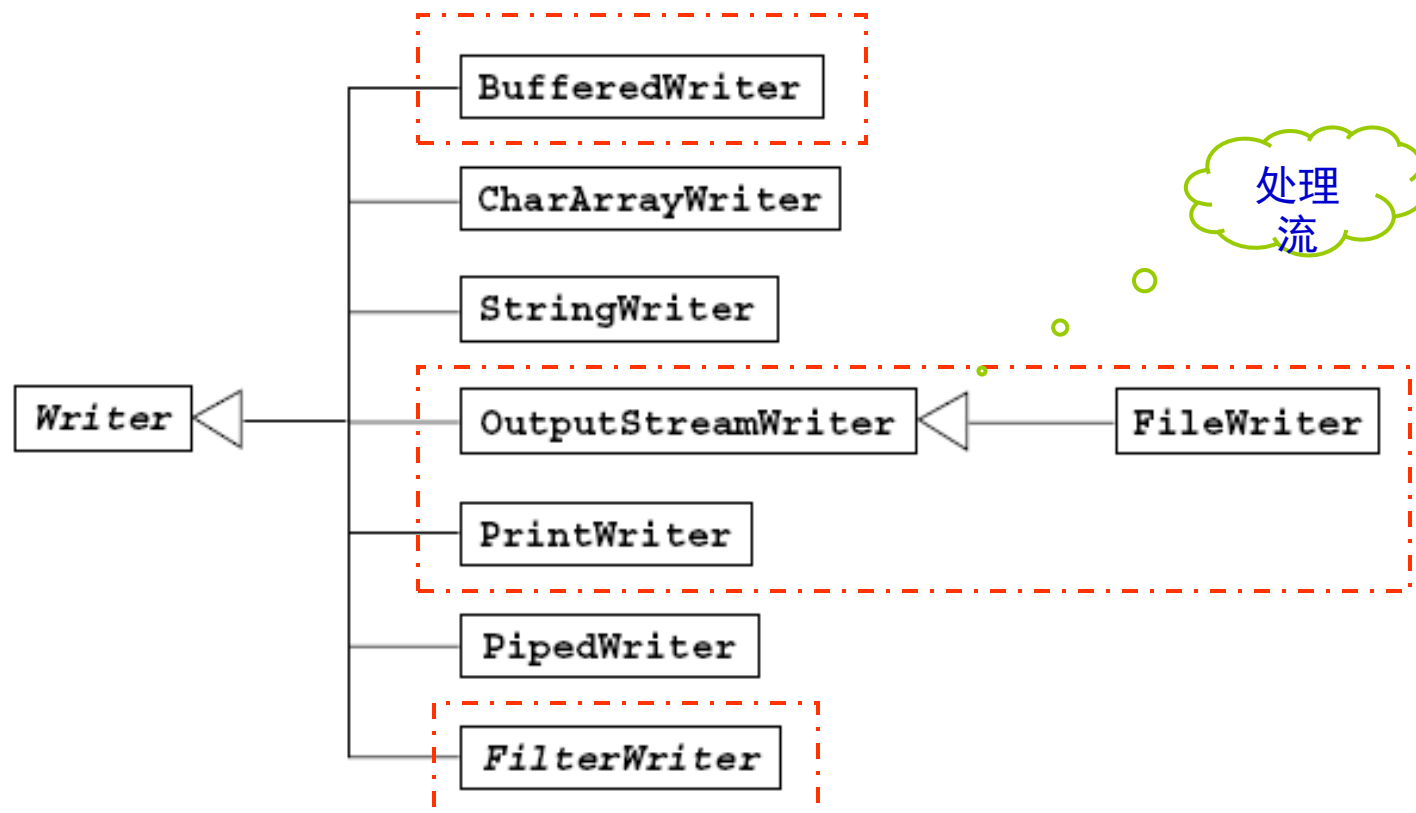
- ✓ 用于输入字符流的抽象类

- ✓ Reader常用的方法

方法	含义
<code>int read()</code>	一次读取一个char的数据，并以int类型把数据返回来，如果没有数据可以读了，会返回” -1”
<code>int read(char[] cbuffer)</code>	把所读取到的数据放在这个char数组中，返回一个int型的数据，这个int型数据存储了返回的真正读取到的数据char数
<code>int read(char[] cbuffer,int offset,int length)</code>	读取length个字符，并存储到一个字节数组cbuffer中，并从offset位置开始返回实际读取的字符数
<code>void close()</code>	关闭此Reader并释放与其关联的所有系统资源

流的概念及API

• 字符输出流的层次结构图



流的概念及API

- 字符输出流的主要方法

- ✓ 输出字符流的抽象类

- ✓ Writer常用的方法

方法	含义
void write(char[] cbuffer)	将要输出的数组先放在一个char数组中，然后用这个方法一次把一组数据输出出去
void write(char[] cbuffer,int off,int len)	将指定字符数组中从偏移量 off 开始的 len 个字符写入此输出流
int write(int b)	将指定的字符写入此输出流
void write(String str)	写入字符串
void write(String str, int off,int len)	将指定字符串中从偏移量 off 开始的 len 个字符写入此输出流
void close()	关闭此输出流并释放与此流有关的所有系统资源
void flush()	刷新此输出流并强制写出所有缓冲的输出字节

节点流与处理流的使用

• 什么是节点流

- ✓ 节点流：从一个特定的数据源（节点）读写数据（如：文件、内存）的类叫做节点流类



- ✓ 这些节点类跟数据源或数据目的地做直接连接用的
- ✓ 在java.io包中，字节继承体系有三种节点类，而字符继承体系有四种节点类

类型	字节流	字符流
File	FileInputStream、FileOutputStream	FileReader、FileWriter
Memory Array	ByteArrayInputStream ByteArrayOutputStream	CharArrayReader CharArrayWriter
Memory String		StringReader、StringWriter
Piped	PipedInputStream PipedOutputStream	PipedReader PipedWriter

节点流的方法

- 节点流的方法 — **InputStream**

方法	含义
<code>int read()</code>	这个方法没有参数，一次读取一个byte的数据，并以int类型把数据返回来，如果没有数据可以读了，会返回” -1”。
<code>int read(byte[] b)</code>	这个方法有一个byte数据类型的参数，这个方法会把所读取到的数据放在这个byte数组中，返回一个int型的数据，这个int型数据存储了返回的真正读取到的数据byte数。
<code>int read(byte[] b,int off,int len)</code>	将输入流中最多 len 个数据字节读入字节,返回值同上。

节点流的方法

- 节点流的方法 — InputStream

方法	含义
<code>void close()</code>	关闭此输入流并释放与该流关联的所有系统资源。
<code>int available()</code>	获取这个流中还有多少个byte的数据可以读取。返回值告诉我们还有多少个byte的数据可以读取。 注：这个方法会产生IOException异常，另外如果InputStream对象调用这个方法的话，它只会返回0，这个方法必须由继承InputStream类的子类对象调用才有作用。
<code>long skip(long n)</code>	跳过和放弃此输入流中的 n 个数据字节。返回值返回真正跳过的字节数。

节点流的方法

- 节点流的方法 — **OutputStream**

方法	含义
<code>void write(byte[] b)</code>	将要输出的数组先放在一个byte数组中，然后用这个方法一次把一组数据输出出去。
<code>void write(byte[] b, int off, int len)</code>	将指定字节数组中从偏移量 off 开始的 len 个字节写入此输出流。
<code>void write(int b)</code>	将要输出的byte数据传给这个方法就可。
<code>void close()</code>	关闭此输出流并释放与此流有关的所有系统资源
<code>void flush()</code>	刷新此输出流并强制写出所有缓冲的输出字节。

节点流的方法

- 节点流的方法 — **OutputStream**

- ✓ 注意：

- 使用write方法输出数据时，有些数据并不会马上输出到我们指定的目的，通常会在内存中有个暂存区，有些输出的数据会暂时存放在这里，如果我们想要立刻把数据输出到目的地，不要放在暂存区中时，可以调用” flush”这个方法对暂存区做清除的动作。
- 同样，数据输出完后，记得把它” close”，在调用close这个方法时，会先调用flush这个方法，以确保所有的数据都已经输出到目的地了。

节点流的方法

- 节点流的方法 — Reader

- ✓ Reader是输入字符数据用的类，它所提供的方法和InputStream类一样，差别在于InputStream类中用的是byte类型，而Reader类中用的是char类型。

- 注：

- Reader类中没有available方法，取而代之的是” ready”方法，这个方法会去检查Reader对象是否已经准备好输入数据了，如果是返回true，反之返回false。

节点流的方法

• 节点流的方法 — Writer

- ✓ Writer类是输出字符数据的类，同样地，提供的方法和OutputStream类中的方法类似，将OutputStream类中用到的byte类型，换成char类型就可。

➤ 注：

- Writer类另外提供了两个writer方法，所以Writer类有5个writer方法，多出来的两个只是把char数据换成String对象而已，方便输出字符的数据。

文件的访问

- 文件的访问

- ✓ 了解了流操作的方法和File类的使用后，我们来看看如何访问一个文件中的数据
- ✓ 在java.io包中，可以利用以下四种节点类来进行文件的访问
 - FileInputStream
 - FileOutputStream
 - FileReader
 - FileWriter

文件的访问

- 文件的访问

- ✓ 了解了流操作的方法和File类的使用后，我们来看看如何访问一个文件中的数据
- ✓ 在java.io包中，可以利用以下四种节点类来进行文件的访问
 - FileInputStream
 - FileOutputStream
 - FileReader
 - FileWriter

文件的访问

- 文件的访问 — **FileInputStream** 示例 **FileInputStreamDemo.java**

- ✓ 构造方法

方法	含义
<code>FileInputStream(String fileurl)</code>	通过打开一个到实际文件的链接来创建一个 <code>FileInputStream</code> ，该文件通过文件系统中的路径名指定
<code>FileInputStream(File fileobj)</code>	通过打开一个到实际文件的连接来创建一个 <code>FileInputStream</code> ，该文件通过文件系统中的 <code>File</code> 对象 <code>file</code> 指定

```
File f = new File("d:/io/a.txt");  
FileInputStream fin1 = new FileInputStream(f);  
FileInputStream fin2 = new FileInputStream("d:/io/b.txt");
```

文件的访问

- 文件的访问 — **FileReader** 示例 **FileReaderDemo.java**

- ✓ 构造方法

方法	含义
<code>FileReader(String fileurl)</code>	通过打开一个到实际文件的连接来创建一个 <code>FileReader</code> ，该文件通过文件系统中的路径名指定
<code>FileReader(File fileobj)</code>	通过打开一个到实际文件的连接来创建一个 <code>FileReader</code> ，该文件通过文件系统中的 <code>File</code> 对象 <code>file</code> 指定

```
File f = new File("d:/io/a.txt");  
FileReader fin1 = new FileReader(file);  
FileReader fin2 = new FileReader("d:/io/b.txt");
```

文件的访问

- 文件的访问 — **FileOutputStream** 示例 **FileOutputStreamDemo.java**

- ✓ 构造方法

方法	含义
<code>FileOutputStream(String fileurl)</code>	创建一个向路径为fileurl的文件中写入数据的输出文件流
<code>FileOutputStream(String fileurl, boolean append)</code>	创建一个向路径为fileurl的文件中写入数据的输出文件流,并将字节写在文件尾处
<code>FileOutputStream(File fileobj)</code>	创建一个向指定 File 对象fileobj表示的文件中写入数据的文件输出流
<code>FileOutputStream(File fileobj,boolean append)</code>	创建一个向指定 File 对象fileobj表示的文件中写入数据的文件输出流,并将字节写在文件尾处

文件的访问

- 文件的访问 — **FileWriter**

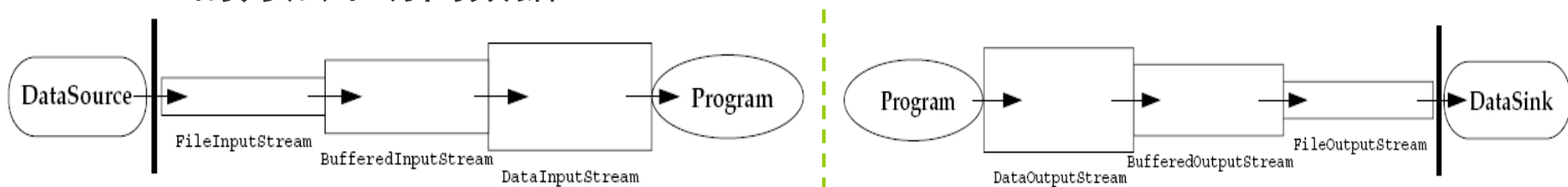
- ✓ 构造方法

方法	含义
<code>FileWriter(String fileurl)</code>	创建一个向路径为fileurl的文件中写入数据的输出文件流
<code>FileWriter(String fileurl, boolean append)</code>	创建一个向路径为fileurl的文件中写入数据的输出文件流,并将字节写在文件尾处
<code>FileWriter(File fileobj)</code>	创建一个向指定 File 对象fileobj表示的文件中写入数据的文件输出流
<code>FileWriter(File fileobj,boolean append)</code>	创建一个向指定 File 对象fileobj表示的文件中写入数据的文件输出流 ,并将字节写在文件尾处

什么是处理流

- 处理流

- ✓ 只用字节或是字符为单位来对数据做输入输出是不够的，有时候我们需要一行一行的读数据，有时我们需要读取特定格式的数据，因此Java提供了这样的机制，能把数据流作连接(chain)，让原本没有特殊访问方法的流，通过连接到特殊的流后，变成可以用特定的方法来访问数据



- ✓ “连接”在已存在的流（节点流或处理流）之上，通过对数据的处理为程序提供更为强大的读写功能
- ✓ 处理流类的构造函数中，都必须接收另外一个流对象作为参数

常见的处理流类

- 处理流类

种类\继承体系	字节	字符
缓冲(Buffered)	BufferedInputStream, BuueredOutputStream	BufferedReader, BufferedWriter
字符和字节转换		InputStreamReader, OutputStreamWriter
对象序列化	ObjectInputStream, ObjectOutputStream	
特定数据类型访问	DataInputStream, DataOutputStream	
计数	LineNumberInputStream	
重复	PushbackInputStream	
打印	PrintStream	PrintWriter

常见的处理流类

- 常见处理流类 — 缓冲流（Buffered）

- ✓ 缓冲流对读写的数据提供了缓冲的功能，提高了读写的效率，同时增加了一些新的方法
- ✓ Java提供了四种缓冲流，其构造方法

构造方法	含义
<code>BufferedInputStream(InputStream in)</code>	创建了一个带有32字节缓冲区的缓冲输入流
<code>BufferedInputStream(InputStream in, int size)</code>	创建了一个带有size大小缓冲区的缓冲输入流
<code>BufferedOutputStream(OutputStream out)</code>	创建了一个带有32字节缓冲区的缓冲输出流
<code>BufferedOutputStream(OutputStream out, int size)</code>	创建了一个带有size大小缓冲区的缓冲输出流

常见的处理流类

- 常见处理流类 — 缓冲流（Buffered）

- ✓ Java提供了四种缓冲流，其构造方法

构造方法	含义
<code>BufferedReader(Reader in)</code>	创建一个使用默认大小输入缓冲区的缓冲字符输入流
<code>BufferedReader(Reader in,int size)</code>	创建一个使用size大小输入缓冲区的缓冲字符输入流
<code>BufferedWriter(Writer out)</code>	创建一个使用默认大小输入缓冲区的缓冲字符输出流
<code>BufferedWriter(Writer out,int size)</code>	创建一个使用size大小输入缓冲区的缓冲字符输出流

常见的处理流类

- 常见处理流类 — 缓冲流 (Buffered)

- ✓ 缓冲流中的方法

- BufferedInputStream支持其父类的mark和reset方法
 - BufferedWriter提供了readLine方法用于读取一行字符串(以\r或\n分隔)
 - BufferedWriter提供了newLine方法用于写入一个行分隔符
 - 对于BufferedOutputStream和BufferdWriter, 写出的数据会先在内存中缓存, 使用flush()方法将使内存中的数据立刻写出

示例 [BufferedStream.java](#) [BufferedRW.java](#)

对象序列化概述

- 对象序列化概述

- ✓ 通过使用ObjectInputStream和ObjectOutputStream类保存和读取对象的机制叫做序列化机制
- ✓ 对象(Object)序列化是指将对象转换为字节序列的过程
- ✓ 反序列化则是根据字节序列恢复对象的过程
- ✓ 序列化一般用于以下场景：
 - 永久性保存对象，保存对象的字节序列到本地文件中
 - 通过序列化对象在网络中传递对象
 - 通过序列化在进程间传递对象

示例 `SerializationDemo.java`

支持序列化的接口和类

- 支持序列化的接口和类

- ✓ 序列化的过程，是将任何实现了 `Serializable` 接口或 `Externalizable` 接口的对象通过 `ObjectOutputStream` 类提供的相应方法转换为连续的字节数据，这些数据以后仍可通过 `ObjectInputStream` 类提供的相应方法被还原为原来的对象状态，这样就可以将对象完成的保存在本地文件中，或在网络 and 进程间传递
- ✓ 支持序列化的接口和类
 - `Serializable` 接口、`Externalizable` 接口
 - `ObjectInputStream`
 - `ObjectOutputStream`

支持序列化的接口和类

- 支持序列化的接口和类

- ✓ **Serializable接口**

- 只有一个实现Serializable接口的对象可以被序列化工具存储和恢复
 - Serializable接口没有定义任何属性或方法。它只用来表示一个类可以被序列化。如果一个类可以序列化，它的所有子类都可以序列化

支持序列化的接口和类

- 支持序列化的接口和类

- ✓ **Externalizable接口**

- 可以让需要序列化的类实现Serializable接口的子接口Externalizable
 - Externalizable接口表示实现该接口的类在序列化中由该类本身来控制信息的写出和读入

支持序列化的接口和类

- 支持序列化的接口和类

- ✓ ObjectOutputStream类

- ObjectOutputStream类继承OutputStream类，并实现了ObjectOutput接口。它负责向流写入对象

- 构造方法

`ObjectOutputStream(OutputStream out)`

- 主要方法

`writeObject(Object obj)`

- ✓ 向指定的OutputStream中写入对象obj

支持序列化的接口和类

- 支持序列化的接口和类

- ✓ **ObjectInputStream类**

- ObjectInputStream类继承InputStream类，并实现了ObjectInput接口。它负责从流中读取对象

- 构造方法

`ObjectInputStream(InputStream in)`

- 主要方法

`readObject(Object obj)`

- ✓ 从指定的InputStream中读取对象

对象序列化的条件

- 对象序列化的条件

- ✓ 该对象类必须实现Serializable接口
- ✓ 如果该类有直接或者间接的不可序列化的基类，那么该基类必须有一个默认的构造器。该派生类需要负责将其基类中的数据写入流中。
- ✓ 建议所有可序列化类都显式声明 serialVersionUID 值。
 - serialVersionUID在反序列化过程中用于验证序列化对象的发送者和接收者是否为该对象加载了与序列化兼容的类。
 - 如果接收者加载的该对象的类的 serialVersionUID 与对应的发送者的类的版本号不同，则反序列化将会导致InvalidClassException。

示例 `SerializationDemo.java` `Customer.java`

transient关键字

- transient关键字

- ✓ transient修饰的属性不进行序列化的操作，起到一定消息屏蔽的效果
- ✓ 被transient修饰的属性可以正确的创建，但被系统赋为默认值。
即int类型为0，String类型为null

□ 注：ObjectInputStream和ObjectOutputStream类不会保存和读写对象中的transient和static类型的成员变量

示例

Customer.java

SerializationDemo.java

本章重点总结

- **文件管理**
 - ✓ File类的使用与文件操作
- **流的概念及API**
 - ✓ 流的概念
- **节点流与处理流的使用**
 - ✓ 节点流
 - ✓ 处理流
- **对象的序列化**
 - ✓ 序列化
 - ✓ transient关键字

课后作业

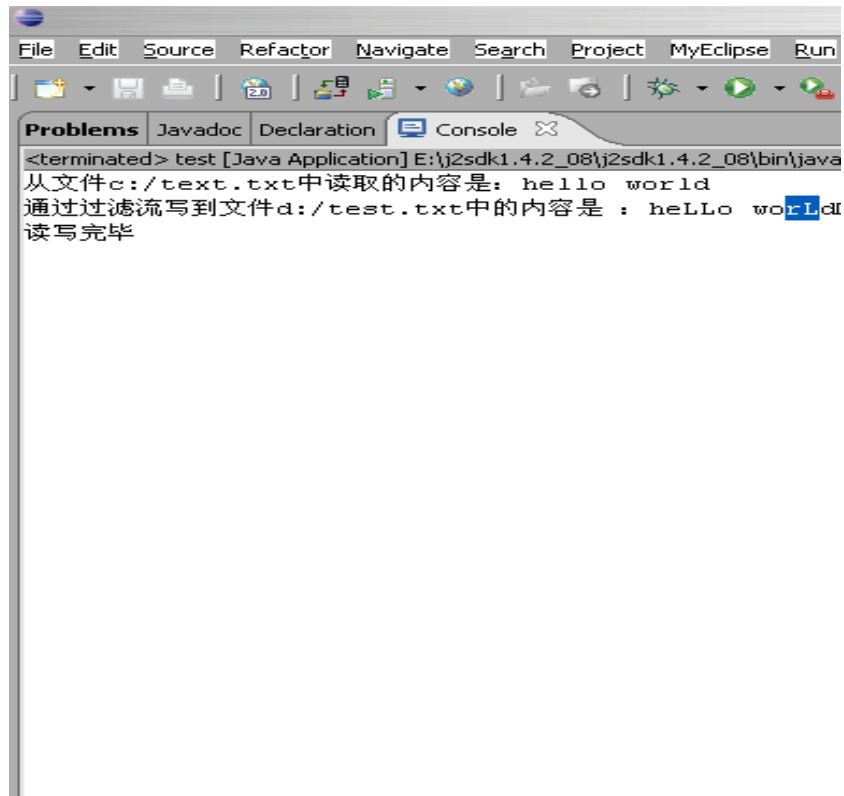
- 1、在本机的磁盘系统中，找一个文件夹，利用File类的提供方法，列出该文件夹中的所有文件的文件名和文件的路径，执行效果如下：[必做题]
- 路径是xxx的文件夹内的文件有：
文件名： abc.txt
路径名： c:\temp\abc.txt

文件名： def.txt
路径名： c:\temp\def.txt
- 2、编写一个java程序实现文件复制功能，要求将d:/io/copysrc.doc中的内容复制到d:/io/copydes.doc中。[必做题]

课后作业

- 3、创建c:/test.txt文件并在其中输入"hello world"
- 创建一个输入流读取该文件中的文本
- 并且把小写的l变成大写L再利用输出流写入到d:\test.txt中 [必做题]
- 3.1 实现步骤：
 - 3.1.1 在本地硬盘C盘下创建一个文件test.txt
 - 3.1.2 创建一个包含main()方法的类，并在main中编写代码
 - 3.1.3 运行代码并且测试结果
- 3.2 实现过滤器的功能
- 效果显示：

课后作业



The screenshot shows the Eclipse IDE's console window. The title bar indicates the application is 'test [Java Application]' located at 'E:\j2sdk1.4.2_08\j2sdk1.4.2_08\bin\java'. The console output consists of three lines of text: '从文件c:/text.txt中读取的内容是: hello world', '通过过滤流写到文件d:/test.txt中的内容是 : heLLo woRLd', and '读写完毕'. The text is displayed in a monospaced font, and the console window has a standard toolbar with icons for file operations and running the application.

```
<terminated> test [Java Application] E:\j2sdk1.4.2_08\j2sdk1.4.2_08\bin\java
从文件c:/text.txt中读取的内容是: hello world
通过过滤流写到文件d:/test.txt中的内容是 : heLLo woRLd
读写完毕
```

课后作业

- 4、在程序中创建一个Student类型的对象，并把对象信息保存到d:/io/student.txt文件中，然后再从文件中把Student对象的信息读出显示在控制台上，Student类的描述如下： [选做题]

Student
-id -name -birth
+toString()