Java技术

第八章 Java I/O 系统

路 强

luqiang@hfut.edu.cn

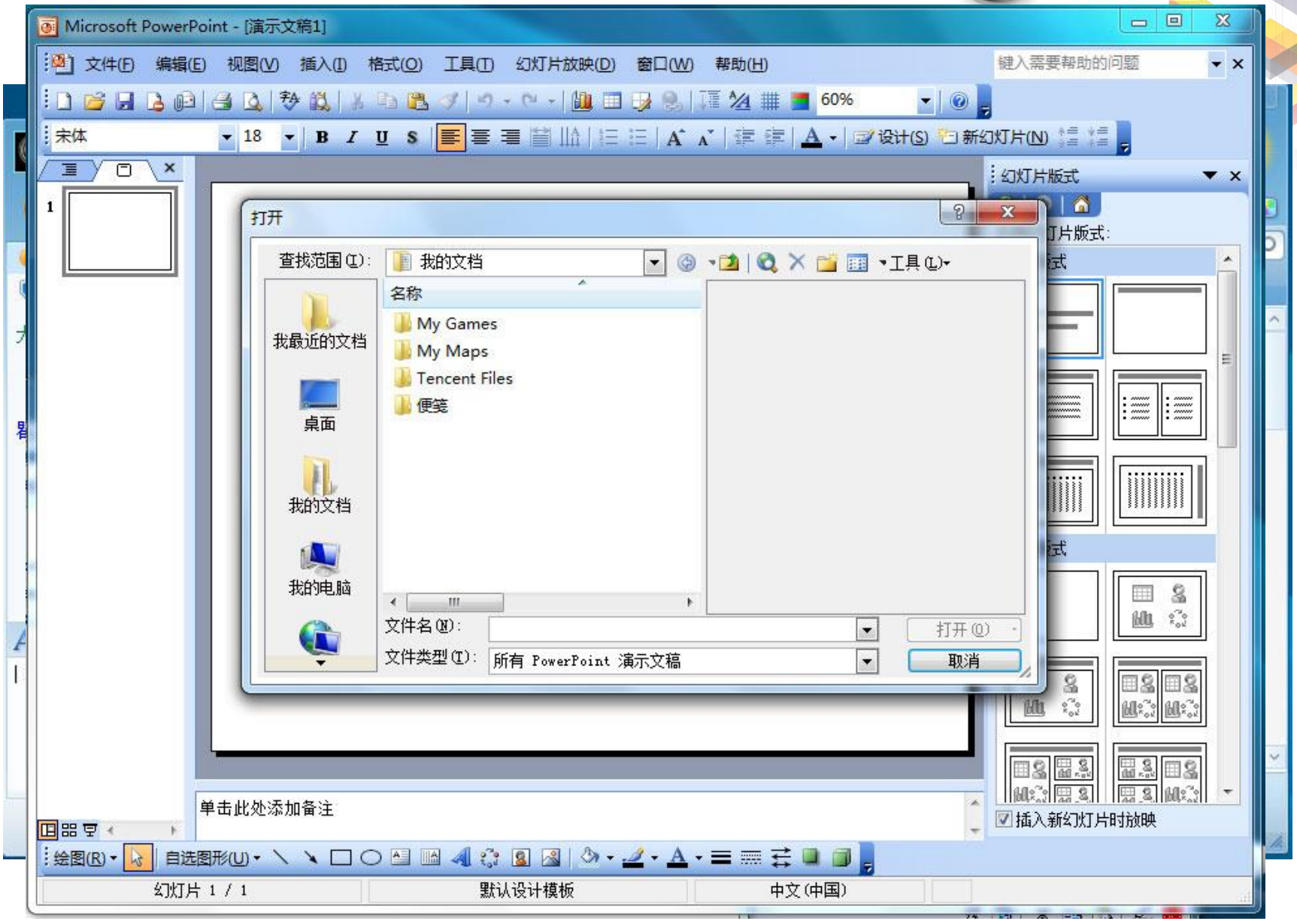
合肥工业大学计算机与信息学院

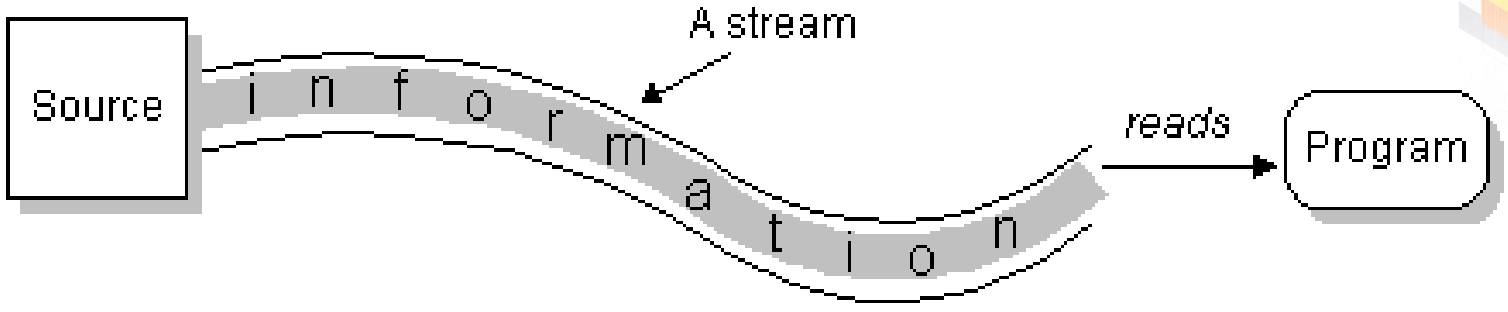
本章学习提示

- 本章我们主要学习Java语言的输入输出处理的机制
  - 文　件的访问
  - 字节流 的使用
  - 字符流 的使用
  - 基本流 的使用

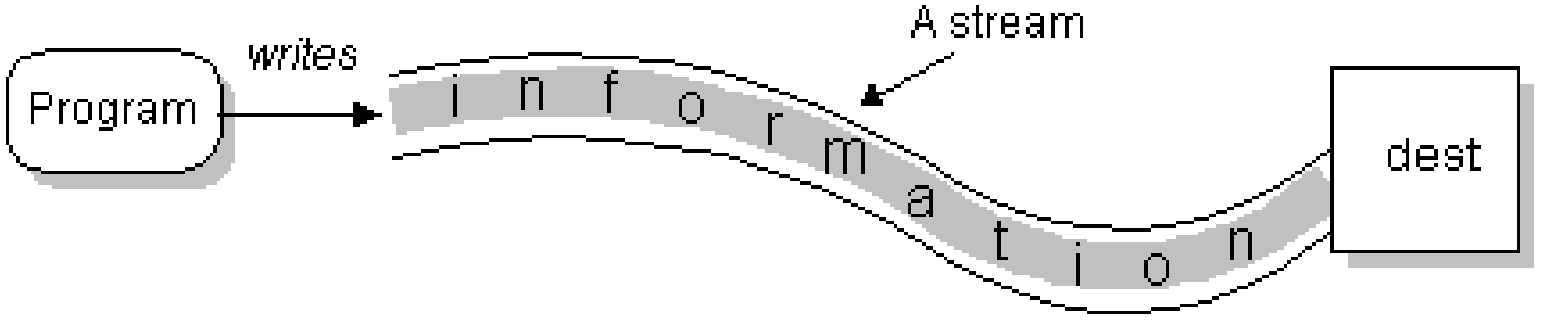
目　录

常用软件


Microsoft PowerPoint - [演示文稿1]
文件(F) 编辑(E) 视图(V) 插入(I) 格式(O) 工具(T) 幻灯片放映(D) 窗口(W) 帮助(H)
键入需要帮助的问题
60%
宋体
18
设计(S) 新幻灯片(N)
幻灯片版式
打开
查找范围(I): 我的文档
工具(L)
名称
My Games
My Maps
Tencent Files
便笺
我最近的文档
桌面
我的文档
我的电脑
文件名(N):
文件类型(T): 所有 PowerPoint 演示文稿
打开(O)
取消
单击此处添加备注
插入新幻灯片时放映
绘图(R)
自选图形(U)
幻灯片 1 / 1
默认设计模板
中文(中国)



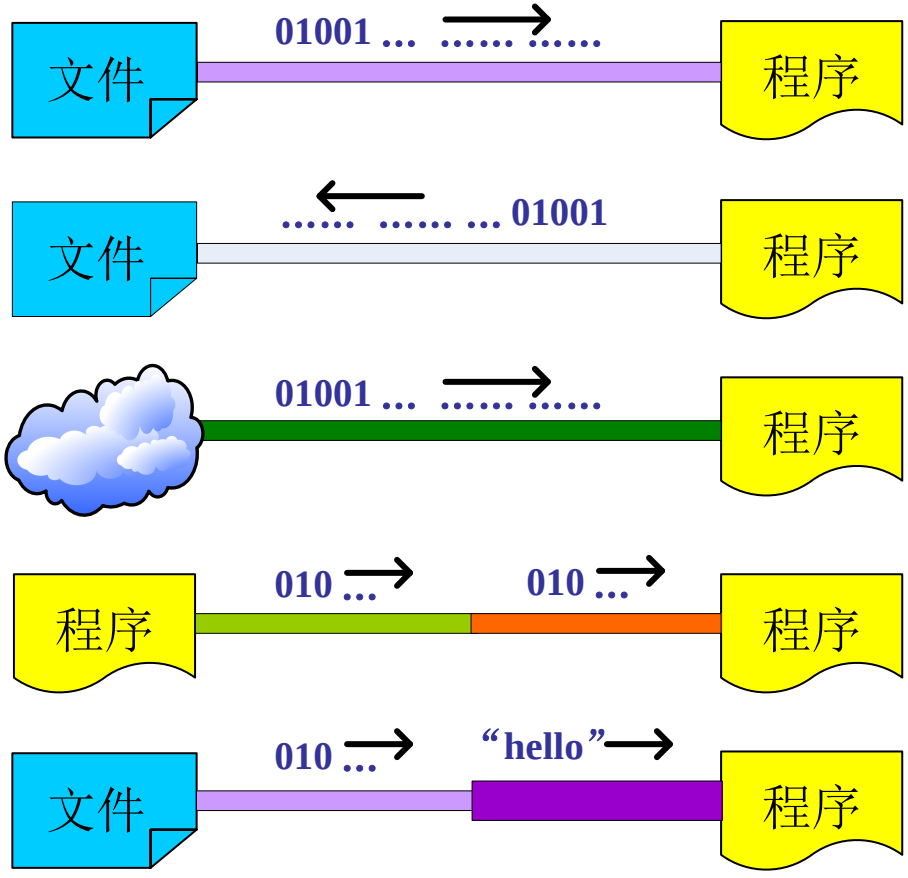
输入流示意图



输出流示意图

输入输出类库

- 输入和输出是**程序与用户之间沟通的桥梁**，程序与用户进行信息交互的重要手段。



输入输出类库

- Java 提供了专用于输入输出功能的包Java.io，其中包含5个非常重要的类，所有与输入输出有关的类都继承了这5个类。
  - 文件流 FileInputStream FileOutputStream
  - 字节流 InputStream OutputStream
  - 字符流 Reader Writer
  - 过滤流 FilterInputStream FilterOutputStream
    - 基本流 System.out System.in
- Java的输入输出是以流（stream)的方式进行处理的。流是在计算机的 输入、输出操作中流动的数据序列。
  Java 按流的单位分有位流（字节流）和字符流；按流动方向分为输入流和输出流。

I/O流的分类

- 按所读写的数据类型分两类：
  - **字节流类**（Byte Streams） 字节流类用于向字节流读写8位二进制的字节。一般地，字节流类主要用于读写诸如图象或声音等的二进制数据。
  - **字符流类**（Character Streams） 字符流类用于向字符流读写16位二进制字符。
  - Java SDK所提供的所有流类型位于java.io包内部，全部继承自以下**四种抽象流类型**。

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

I/O流的分类

- 可以**直接**和“信息源”或“信息去向”目标关联的流对象

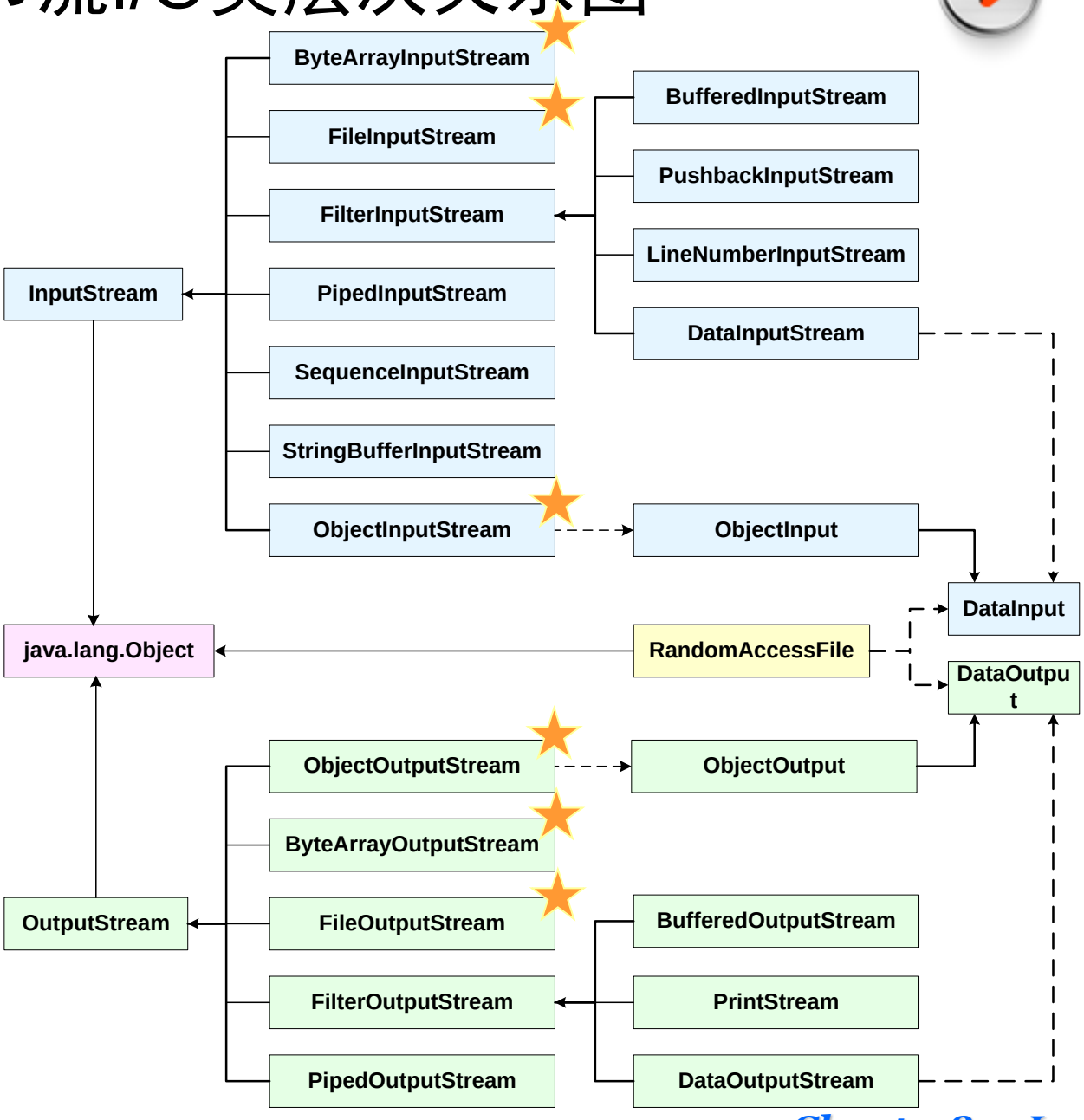
类　型	字 符 流	字 节 流
File（文件）	FileReader FileWriter	FileInputStream FileOutputStream
Memory Array	CharArrayReader CharArrayWriter	ByteArrayInputStream ByteArrayOutputStream
Memory String	StringReader StringWriter	—
Pipe（管道）	PipedReader PipedWriter	PipedInputStream PipedOutputStream

目　录

字节流

- 输入流类InputStream和输出流OutputStream是两个最基本的输入输出抽象类，用于处理字节流。
- InputStream类
  - 类中包含了一套所有输入流都需要的方法，可以完成最基本的从输入流读取数据的功能。
  - 当Java程序需要从外设中读入数据时，先创建一个适当类型的输入流类对象来完成与外设的连接，然后再调用执行该新建对象的特定方法，实现对相应外设的操作。
- 每次执行时都从输入流的当前位置处读入**一个字节的二进制数据**，读数作为低字节，高字节配全零，合成为一个整型量返回，若输入流当前位置无数据**返回-1**。

字节流I/O类层次关系图



InputStream类的常用子类

- 类中包含了一套所有输入流都需要的方法，可以完成最基本的从输入流读取数据的功能。
- 当Java程序需要从外设中读入数据时，先创建一个适当类型的输入流类对象来完成与外设的连接，然后再调用执行该新建对象的特定方法，实现对相应外设的操作。
- InputStream 子类对象继承InputStream 类的如下方法
  - read()方法、定位指针的方法、close()方法。
  - **public int read()**
    - 每次执行时都从输入流的当前位置处读入一个字节的二进制数据，读数作为低字节，高字节配全零，合成为一个整型量返回，若输入流当前位置**无数据返回 -1**。
  - 使用完毕后，应用close()方法关闭流

InputStream类的常用子类

- FileInputStream

  用于从文件File中将诸如图像数据之类的原始字节的数据流读出

- ByteArrayInputStream

  此类实现了一个输出流，从一个 byte 数组中读出数据

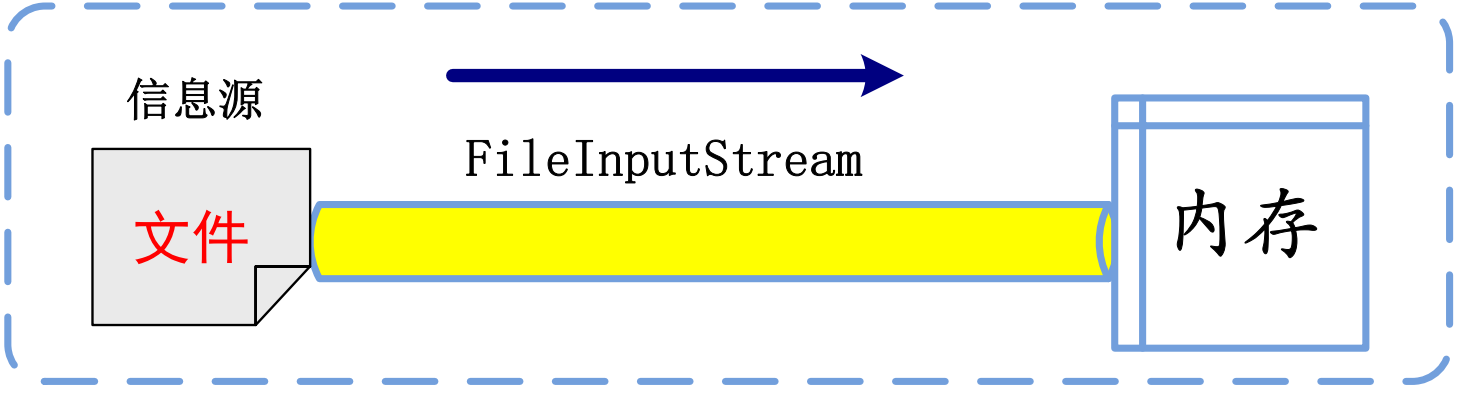
- ObjectInputStream

  实现对象的持久存储，从 OutputStream读入Java 对象的基本数据类型和图形

详细内容见
软件包 java.io
的InputStream类

实　例

- 使用FileInputStream类，从文件中以字节流方式读取全部信息在命令行打印输出



OutputStream类

- 类中包含所有输出流都要使用的方法。

  当Java程序需要向某外设输出数据时，先要创建一个输出流类对象，通过该对象实现与外设的连接，再利用OutputStream类提供的方法将数据写入该外设中。

- OutputStream类是一个抽象类，不能直接创建OutputStream类对象，而应该创建它的某个子类的对象。

  子类继承的方法有：
  - Write()方法、flush()方法和close()方法
  - public void write(int c);　将参数c对应的字符写入输出流
  - public void write(char array[]);

    将字符数组array中全部字符顺序写入到输出流
  - 使用完毕后，应用close()方法关闭流

OutputStream类的常用子类

- FileOutputStream

  用于将诸如图像数据之类的原始字节的数据流写入 File

- ByteArrayOutputStream

  此类实现了一个输出流，数据被写入一个 byte 数组

- ObjectOutputStream

  实现对象的持久存储，将 Java 对象的基本数据类型和图形写入 OutputStream

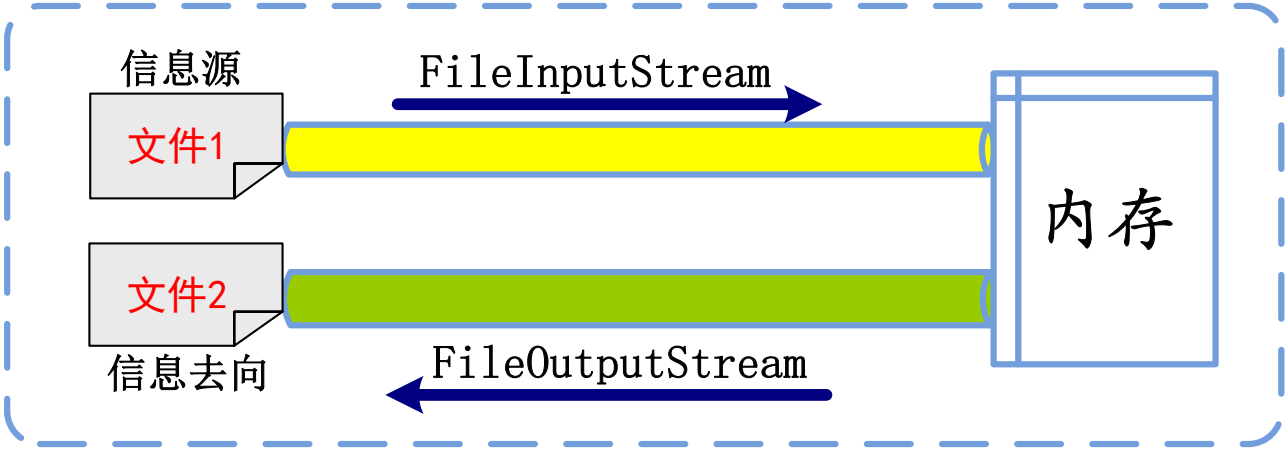
- PipedOutputStream

  将管道输出流连接到管道输入流来创建通信管道

详细内容见
软件包 java.io
的OutputStream类

实　例 2

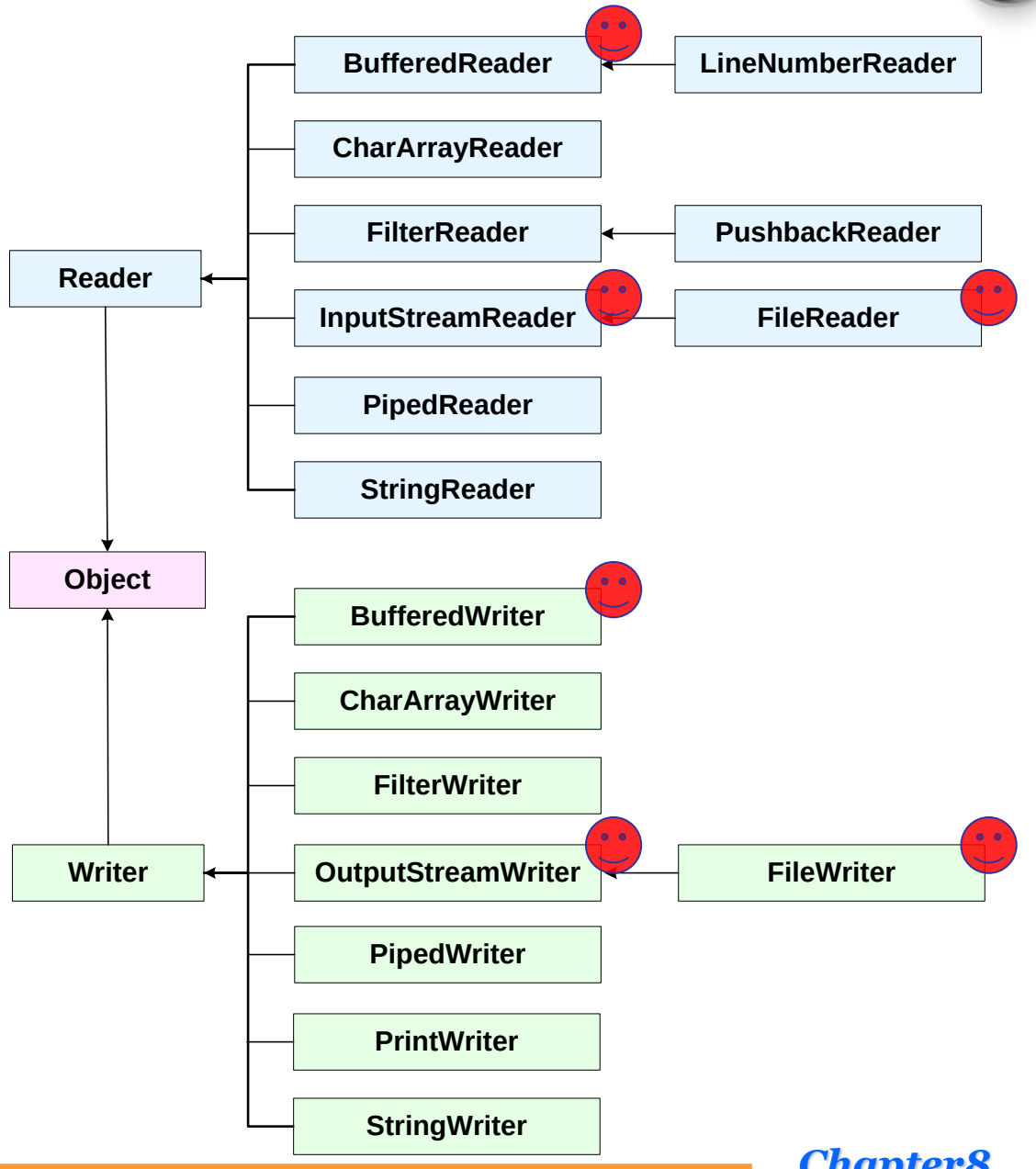
- 使用FileInputStream类和FileOutputStream类，以字节流方式将文件1中信息复制到文件2中



TestFileCopy_Stream.java

目　录

Unicode字符流I/O类层次关系图



Reader类

- 用来以字符方式从流中读入数据。 Reader类中包含了一套所有字符输入流都需要的方法，可以完成最基本的从字符输入流读取数据的功能。
- Reader是一个**抽象类**，所以实际应用中创建的对象是Reader某个子类的对象，通过该子类对象与外接数据源连接。
- 子类必须实现的方法只有
  - **int read(char[ ] b, int offset, int length)**
  - **void close()**
- 当输入流使用完毕后，可以调用该方法将其关闭，断开Java程序与外设数据源的联系，释放此连接所占用的系统资源

Reader常用子类

- BufferedReader
  - 从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取
  - 可以指定缓冲区的大小
- InputStreamReader
  - 字节流通向字符流的桥梁：它使用指定的 charset 读取字节并将其解码为字符。
  - 每次调用 InputStreamReader 的read() 方法都会导致从底层输入流读取一个或多个字节。
- FileReader
  - 用来读取**字符文件**的类
  - 用于读取**字符流**

详细内容见
软件包 java.io
的Reader类

Writer类

- 用来**以字符方式**向输出流中写入数据。

  Writer类中包含了一套所有字符输出流都需要的方法，可以完成最基本的向字符输出流写入数据的功能。

- Writer是一个**抽象类**，所以实际应用中创建的对象是**Writer**某个子类的对象，通过该子类对象与外接数据源连接。

- 子类必须实现的方法仅有

  - **void write(char[] b, int offset, int length)**
  - **void close()**
  - **void flush()　　//强制清空缓冲区**

- 当输入流使用完毕后,关闭输出流，断开Java程序与外设数据源的连接，释放所占有的系统资源

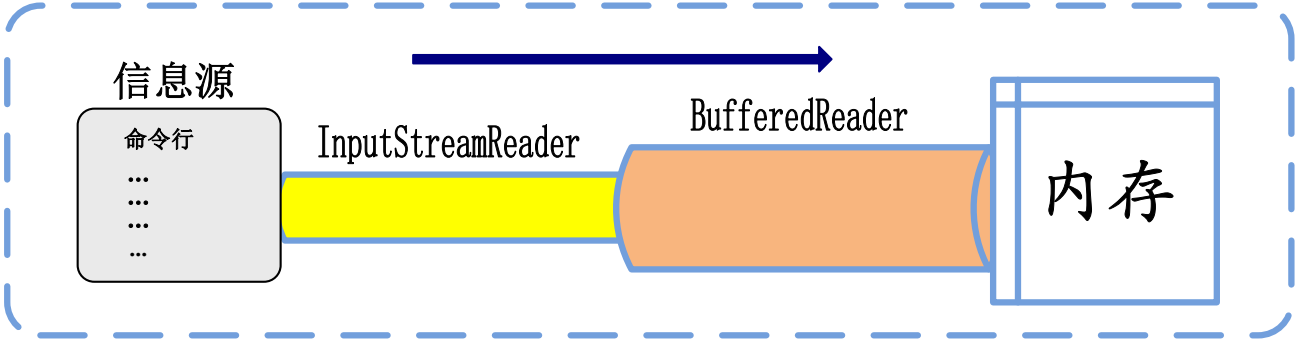
Writer常用子类

- BufferedWriter
  - 将文本写入字符输出流，缓冲各个字符
  - 可以指定缓冲区的大小
- OutputStreamWriter
  - 字符流通向字节流的桥梁：可使用指定的 charset 将要写入流中的字符编码成字节
  - 每次调用 write() 方法都会导致在给定字符（或字符集）上调用编码转换器
- FileWriter
  - 用来写入字符文件的类
  - 文件是否可用或是否可以被创建取决于底层平台
- PrintWriter
  - 向文本输出流打印对象的格式化表示形式

详细内容见
软件包 java.io
的Writer类

以字符流方式读入 – 例程

1．从命令行读入字符串



字符流之例

```
1.  //输入字符串、浮点数、整数在屏幕上显示输入结果
2.  import  java.io.*;
3.  public  class standardIO3{
4.      public static void main(String[] args) throws IOException{
5.          InputStreamReader iin=new InputStreamReader(System.in);
6.          BufferedReader bin=new BufferedReader(iin);
7.          String  s;
8.          float   f;
9.          int i=0;
10.         System.out.println("输入任一字符串");
11.         s=bin.readLine();
12.         System.out.println("输入浮点数");
13.         f=Float.parseFloat(bin.readLine());
14.         System.out.println("输入整数");
15.         i=Integer.parseInt(bin.readLine());
16.         System.out.println("输入的字符串："+s);
17.         System.out.println("输入的浮点数："+f);
18.         System.out.println("输入的整数："+i);
19.     }
20. }
```

字节流->字符流

自己课后分析！

目　录

文件处理

- 在程序中要对磁盘文件或目录进行操作，首先要对文件或目录建立连接，为此Java提供了File类。
- File类位于java.io包中，但不是流类，它不负责输入或输出，而专门用来管理磁盘文件和目录。
- 类的构造方法
  - FileInputStream(String fileName)

    参数fileName表示带路径的磁盘文件名
  - FileInputStream(File file)

    参数file表示为磁盘文件所建立的File对象名
- 在程序执行过程中会出现文件找不到或读写错误，因此程序中必须要对异常进行捕获和处理

用File类访问磁盘文件

问　题

- 如何获取磁盘上的文件和文件夹？

FileList.java

File类

- File(String pathname)
  通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例。
- exists()
  测试此抽象路径名表示的文件或目录是否存在。
- canRead()
  测试应用程序是否可以读取此抽象路径名表示的文件。
- 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表示的目录中的文件。
- mkdir()
  创建此抽象路径名指定的目录。

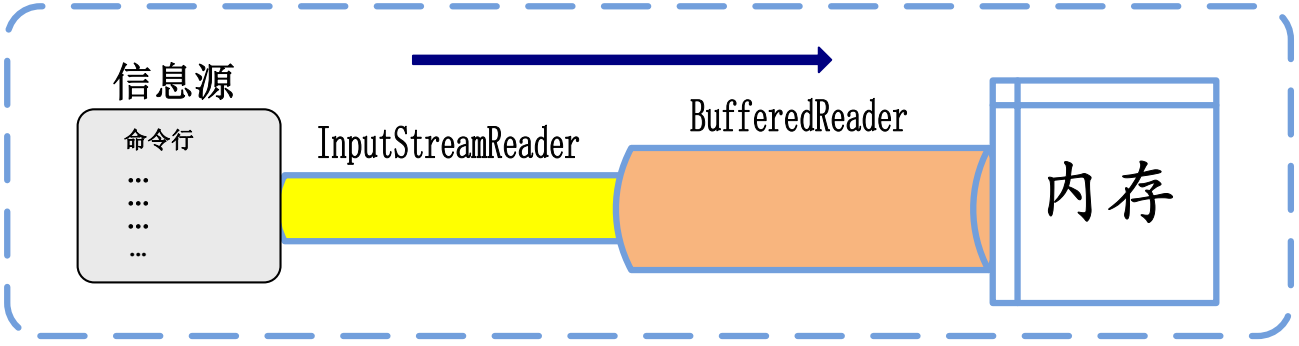
以字符流方式写入文件

- 以字符流方式向文件写入或从文件读出数据，可以使用Writer和Reader类及其子类。它们是抽象类，不能实例化，只能通过它们的子类对象对文件进行操作。
  - 常用的Writer类的子类有FileWriter类和BufferedFileWriter类。
- FileWriter类构造方法
  - FileWriter(String fileName);
    参数fileName表示带路径的磁盘文件名
  - FileWriter(File file);
    参数file表示为磁盘文件所建立的File对象名
- 使用BufferWriter类的方法是
  1. 为文件建立FileWriter对象
  2. 再为该FileWriter对象建立BufferWriter类对象
  3. 写入操作将使用所建立的BufferWriter类对象

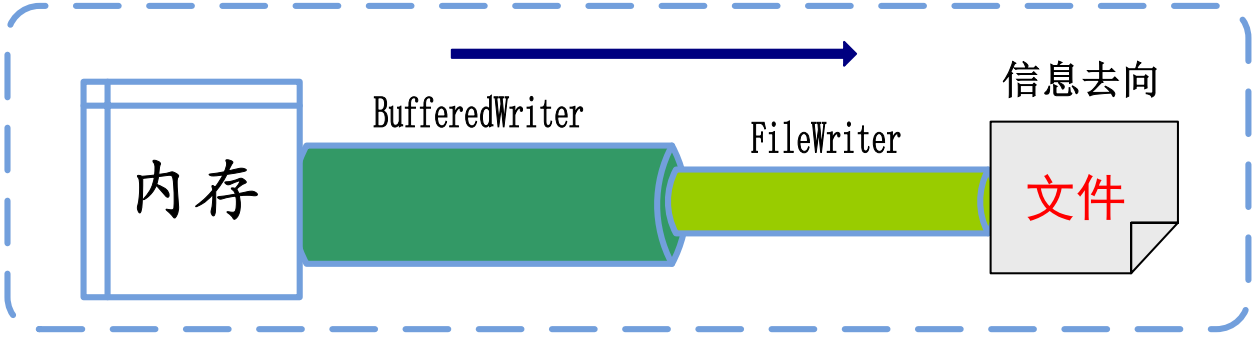
以字符流方式写入文件 – 例程

- 以字符流的方式从命令行读入多行字符，并按行写入磁盘文件 **e:\\dataFile.txt**

1．从命令行读入字符串



2．将字符串写入到磁盘文件



字符流的方式写入磁盘文件


```
1.  //以字符流的方式写入磁盘文件c:\\dataFile.txt中
2.  import java.io.*;
3.  public class FileIO3_write{
4.      public static void main(String args[]) throws IOException{
5.          InputStreamReader iin=new InputStreamReader(System.in);
6.          BufferedReader br=new BufferedReader(iin);
7.          FileWriter fw1=new FileWriter("c:\\dataFile.txt");
8.          BufferedWriter bw=new BufferedWriter(fw1);
9.          String s;
10.         System.out.println("将输入的字符串写入dataFile.txt文件");
11.         while(true){
12.             System.out.println("输入一个字符串: ");
13.             System.out.flush();// 清空输出缓冲区
14.             s=br.readLine();   //写入换行符
15.             if(s.length()==0)break;
16.             bw.write(s);
17.             bw.newLine();
18.         }
19.         bw.close();
20.     }
21. }
```

自己课后分析！！

FileIO3_write.java

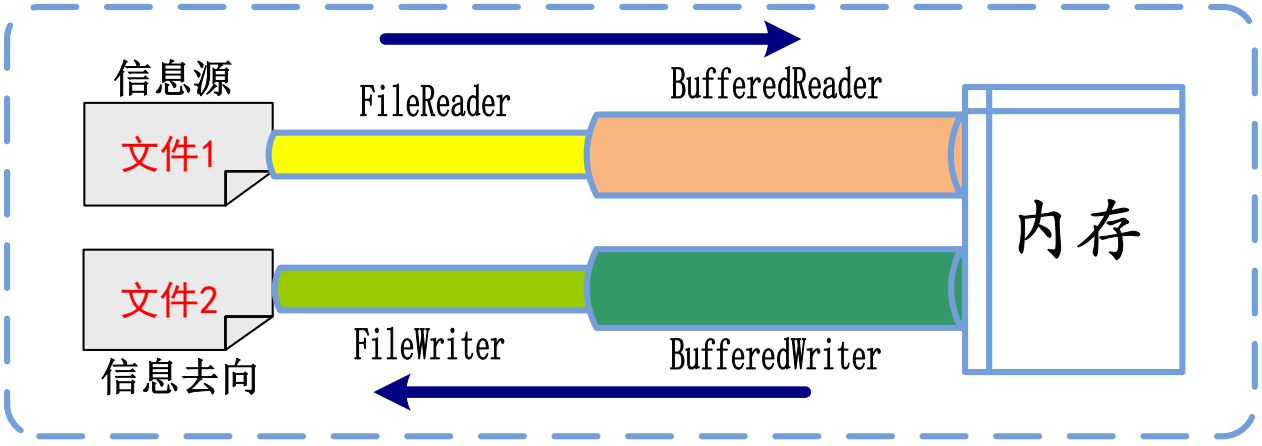
以字符流的方式向显示器输出


```
1.  //本程序的功能是以字符流的方式向显示器输出，
2.  //将 c:\\dataFile.txt数据显示到屏幕上
3.  import java.io.*;
4.  public class FileIO3_read{
5.     public static void main(String args[])throws IOException{
6.        FileReader fr1=new FileReader("c:\\dataFile.txt");
7.        BufferedReader br1=new BufferedReader(fr1);
8.        BufferedWriter bw1=new BufferedWriter(new OutputStreamWriter(System.out));
9.        int lineNum=0;
10.       String s=br1.readLine();
11.       System.out.println("输入文件是：c:\\dataFile.txt");
12.       while(s!=null){
13.         lineNum++;
14.         bw1.write(String.valueOf(lineNum));
15.         bw1.write("    ");
16.         bw1.write(s);
17.         bw1.newLine();
18.       //  System.out.println(s);
19.         s=br1.readLine();
20.        }
21.       bw1.close();
22.    }
23. }
```

自己课后分析！！

FileIO3_read.java

以字符流方式转存文件 – 例程



- 使用BufferedReader类 和 FileReader类 从信息源文件中读取内容
- 使用BufferedWriter类 和 FileWriter类向目标文件中写文件
- “读文件”和“写文件” 以 “行” 为单位

➢ 应用场景：读入日志log文件进行分析

FileIO3_restore.java

字符流的方式转存文件

```
1.  //以字符流的方式读取文件，
2.  //将c:\\dataFile.txt数据备份到targetFile.txt中
3.  import java.io.*;
4.  public class FileIO3_restore {
5.     public static void main(String args[])throws IOException{
6.        FileReader fr1=new FileReader("c:\\dataFile.txt");
7.        BufferedReader br1=new BufferedReader(fr1);
8.        BufferedWriter bw1=new BufferedWriter(new FileWriter("c:\\targetFile.txt"));
9.        int lineNum=0;
10.       String s=br1.readLine();
11.       System.out.println("输入文件是：c:\\dataFile.txt");
12.       System.out.println("输出文件是：c:\\targetFile.txt");
13.       while(s!=null)   { // 每次处理一行！
14.         lineNum++;
15.         bw1.write(String.valueOf(lineNum));
16.         bw1.write("    ");
17.         bw1.write(s);
18.         bw1.newLine();
19.         s=br1.readLine();
20.        }
21.       bw1.close();
22.    }
23. }
```

FileIO3_restore.java

向磁盘文件写入各类数据

- Java 通过DataInputStream & DataOutputStream 类实现各类数据的读写。向文件写入各种类型数据的具体步骤是：
  1. 为磁盘文件建立File类对象
  2. 为File对象建立FileOutputStream类流对象，建立其与磁盘文件的连接
  3. 为FileOutputStream类流对象建立DataOutputStream类对象
  4. 利用DataOutputStream 类的writeInt(),writeFloat(), writeDouble() ,writeBoolean() 等方法，分别向文件中写入整型、单精度型、双精度型、布尔型等数据
  5. 写入操作完成后，利用close()方法将流关闭，断开与磁盘文件的联系

选修内容

将数据写入磁盘文件


```
1.  import java.io.*;
2.    public class FileIO2_write   {
3.     public static void main(String args[])throws IOException     {
4.        int ch;
5.        InputStreamReader iin=new InputStreamReader(System.in);
6.        BufferedReader bin=new BufferedReader(iin);
7.        File file1=new File("c:\\dataInFile.txt");
8.        try {       FileOutputStream fout=new FileOutputStream(file1);
9.                    DataOutputStream dout=new DataOutputStream(fout);
10.                   System.out.println("输入整数 ");
11.                   int i=Integer.parseInt(bin.readLine());
12.                   System.out.println("输入浮点数 ");
13.                    float f=Float.parseFloat(bin.readLine());
14.                   System.out.println("输入布尔量 ");
15.                   boolean b=new Boolean(bin.readLine()).booleanValue();
16.                   System.out.println("输入结果在c:\\dataInFile.txt文件中 ");
17.                   dout.writeInt(i);  dout.writeFloat(f);  dout.writeBoolean(b);
18.                   dout.close();
19.       }  catch(FileNotFoundException e)  {
20.             System.out.println(e);
21.       }  catch(IOException e)  {
22.             System.out.println(e); }
23.    }  }
```

从磁盘文件读出数据并转存


```
1.  import java.io.*;
2.  public class FileIO2_read  {
3.     public static void main(String args[])throws IOException    {
4.        int ch;
5.        File file1=new File("c:\\dataInFile.txt");
6.        File file2=new File("c:\\dataOutFile.txt");
7.        try {
8.           FileInputStream fin=new FileInputStream(file1);
9.           DataInputStream din=new DataInputStream(fin);
10.          int i=din.readInt();     float f=din.readFloat();
11.          boolean b=din.readBoolean();        din.close();
12.          FileOutputStream fout=new FileOutputStream(file2);
13.          DataOutputStream dout=new DataOutputStream(fout);
14.          dout.writeInt(i);    dout.writeFloat(f);    dout.writeBoolean(b);
15.          dout.close();
16.          System.out.println("输入文件是c:\\dataInFile.txt  " + "输出文件是c:\\dataOutFile.txt");
17.          System.out.println("整数："+i);    System.out.println("浮点数 "+f);
18.          System.out.println("布尔量 "+b);
19.       } catch(FileNotFoundException e)  {
20.          System.out.println(e);
21.       } catch(IOException e)  {
22.          System.out.println(e);}
23.    }
24. }
```

目　录

基本流

- 为了减少程序开发人员，因频繁应用标准的输入输出设备，需要频繁地建立输入输出流对象的工作量，java系统**预先定义好3个流对象**，分别表示标准输出设备、标准输入设备和标准错误设备。他们分别是：
  - System.in ：用于程序的输入；　　对应外设为键盘
  - System.out：用于一般输出；　　对应外设为屏幕
  - System.err：用于显示出错信息；　对应外设为屏幕
- System 类的所有属性都是**静态static**的，调用时以类名System为前缀。

  上述3个流对象均为静态属性。

基本I/O之例


```
1.  //从键盘输入字符，然后在屏幕上显示输入结果
2.  import  java.io.*;
3.  public  class standardIO{
4.      public static void main(String[] args) throws IOException {
5.      //IO操作必须捕获IO异常
6.      char c;
7.      System.out.println("输入任一字符 ");
8.      c=(char)System.in.read();
9.      System.out.println("输入的字符是 ： "+c);
10.     }
11. }
```

基本I/O之例


```
1.  //基本IO standardIO2.java，从命令行读入字符串，并处理和显示
2.  import java.io.*;
3.  public class standardIO2{
4.      public static void main(String[] args) throws IOException {
5.          //IO操作必须捕获IO异常
6.          //先使用System.in构造InputStreamReader，再构造BufferedReader。
7.          InputStreamReader iin = new InputStreamReader(System.in);
8.          BufferedReader stdin=new BufferedReader(iin);
9.          //读取并输出字符串。
10.         System.out.print("Enter a string: ");
11.         System.out.println(stdin.readLine());
12.         //读取字符串并转换成double类型数据输出
13.         System.out.print("Enter a double: ");
14.         //将字符串解析为带符号的double类型数据。
15.         double number2=Double.parseDouble(stdin.readLine());
16.         System.out.println(number2);
17.     }
18. }
```

字节流->字符流

本章总结

- 本章我们主要学习Java语言的输入输出处理的机制
  - 基本流的使用
  - 字符流的使用
  - 字节流的使用
  - 文　件 的访问

作业4

作业4：见作业文档。


Thank You !