第8章 输入输出数据流

南京农业大学

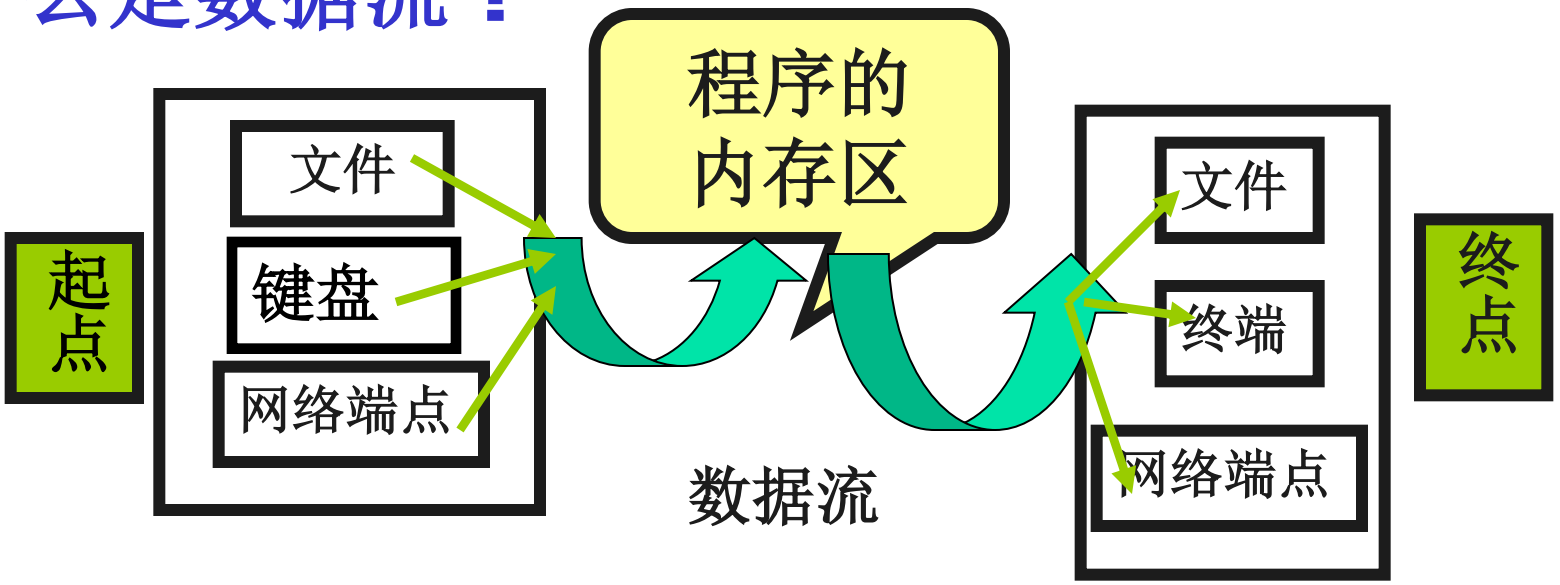
谢忠红

输入输出数据流

- （1）数据流的基本概念
- （2） **java.io**包中的输入输出流的基类
- （3）基本输入输出流的子类
- -文件流
- -缓冲输入输出流
- （4）**IOException**异常类的子类
- （5） 随机存取文件
- （6）输入输出流的高级应用管道流和数据流

（1）数据流的基本概念

什么是数据流？



- 数据流是指所有的**数据通信通道**
- 包括输入流和输出流。

Java的标准的数据流

- java.lang.System
- public final class System extends Object{
- static PrintStream err;//标准错误流（输出）
  static InputStream in;//标准输入(*键盘输入流*)
- static PrintStream out;//标准输出流(*显示器输出流*)
- }

- 注意：
- （1）System类不能创建对象，只能直接使用它的三个静态成员。
- （2）每当main方法被执行时,就自动生成上述三个对象。

- **System.out:**
- 把输出送到缺省的显示(通常是显示器),
  - **Void print(参数)**
  - **Void println(参数)**

- **System.in**
- 从标准输入获取输入(通常是键盘)
- **int read()**
  - //返回**ASCII**码。若,返回值=-1， 说明没有读取到任何字节读取工作结束。
- **int read(byte[] b)**
  - //读入多个字节到缓冲区**b**中返回值是读入的字节数

- **System.err**
- 把错误信息送到缺省的显示

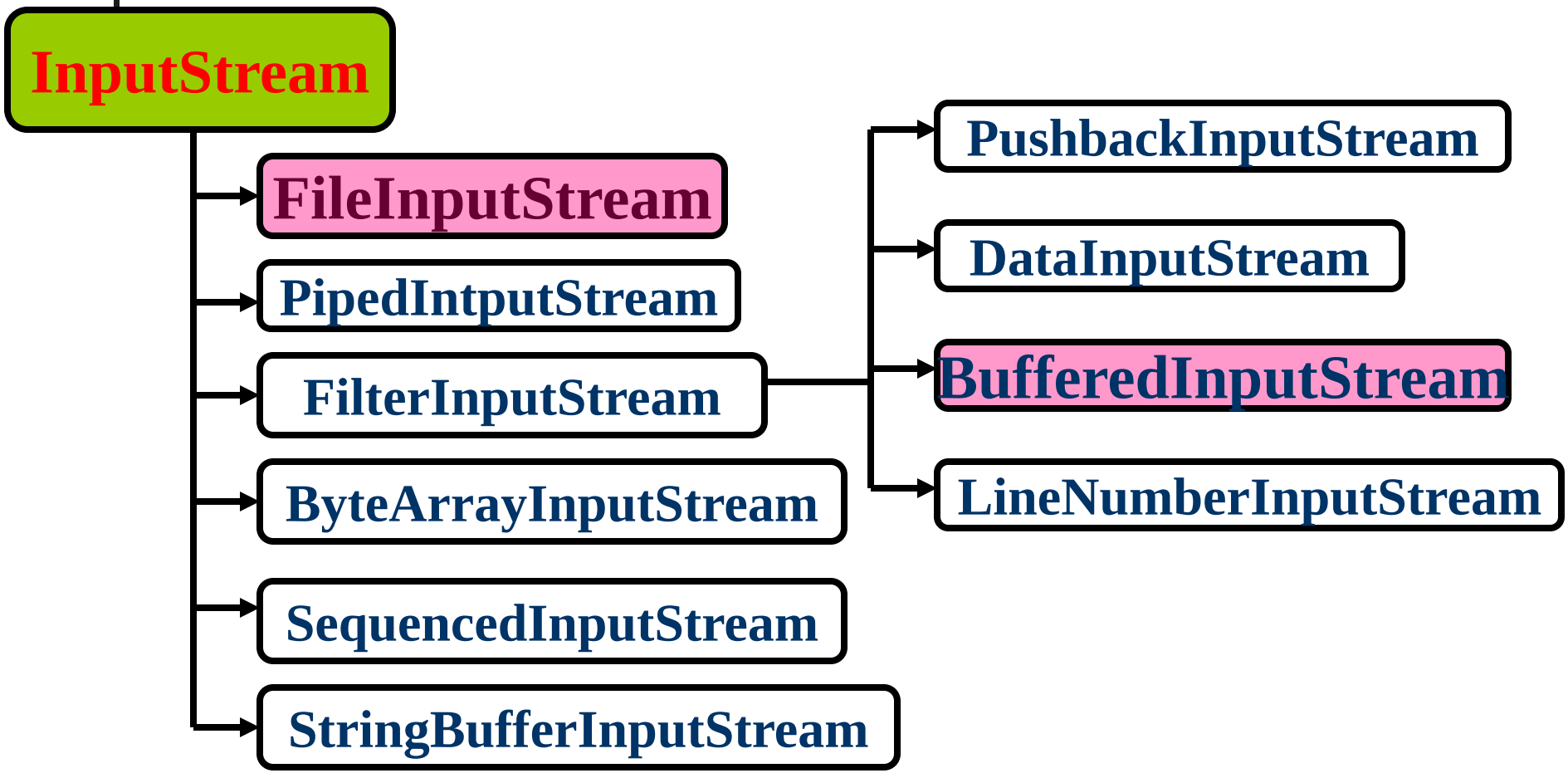
(2) java.io包中的输入输出流的基类

- 输入流:将数据从外界(*键盘、文件、网络...*)
- 读入到程序的通信通道.
- 输入流的基类: **InputStream**

输出流:将程序中的数据输出到外界（显示器、打印机、文件、网络...）的通信通道。

输出流的基本类: **OutputStream**

InputStream 及 其子类

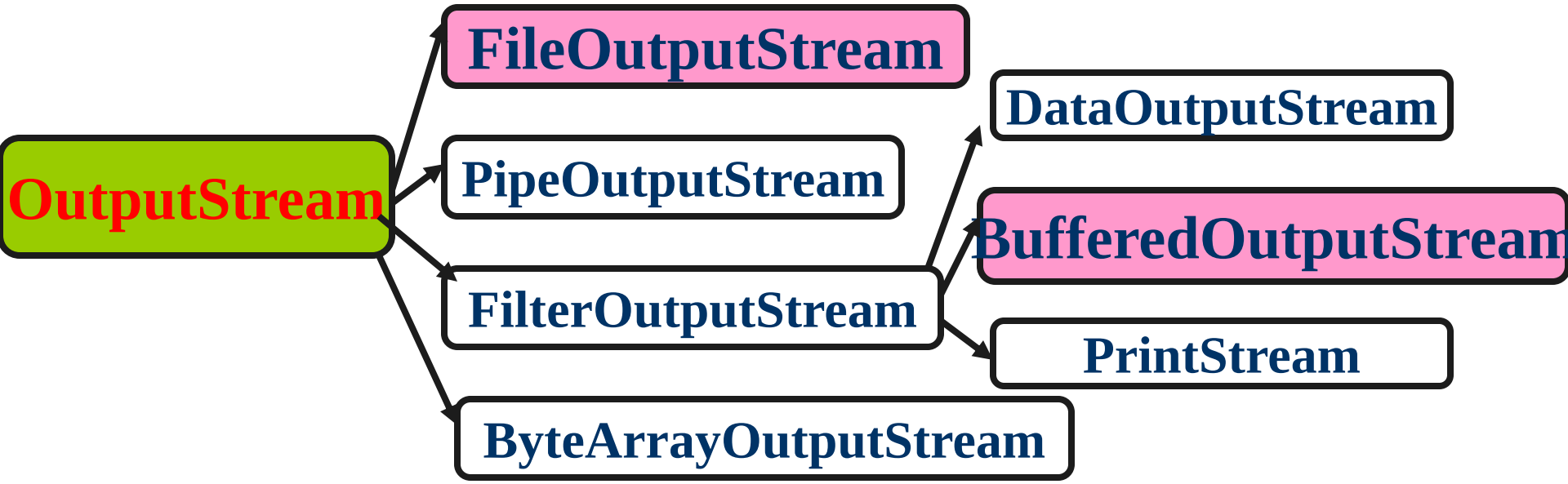


抽象类 InputStream

- **InputStream类的常见方法:**
- **int read():读一个byte类型数,返回读入字节。若返**
- **回值=-1，说明没有读取到任何字节读取工作结束**
- **int read(byte b[]):读多个字节到数组b中**
- **int read(byte b[] ,int off,int len);从输入流中读取**
- **len个字节放入数组b中，起始位置off**
- **available():返回输入流中可以读取的字节数**

close():关闭流

抽象类OutputStream及其子类



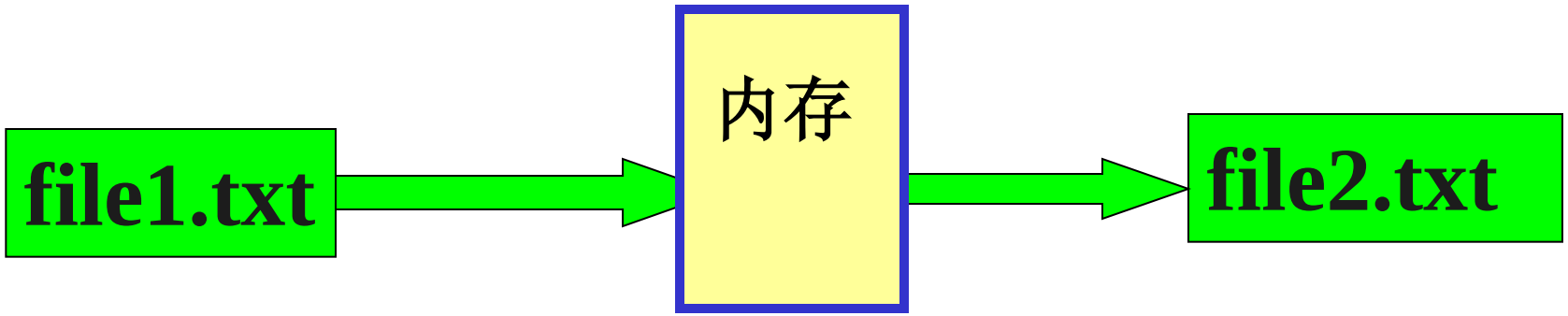
抽象类OutputStream

- **OutputStream类的常见方法:**
- **void write(int b):** 将一个整数输出到流中
- **void write(byte b[]):** 将数组中的数据输出到流中。
- **void write(byte b[], int off,int len):** 将数组b中从off指定的位置开始len长度的数据输出到流中
- **close():** 关闭流.

注意: 上述各方法都有可能引起异常。

（3）基本输入输出流的子类—— 文件流

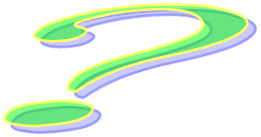
进行文件输入和输出的*通信通道*

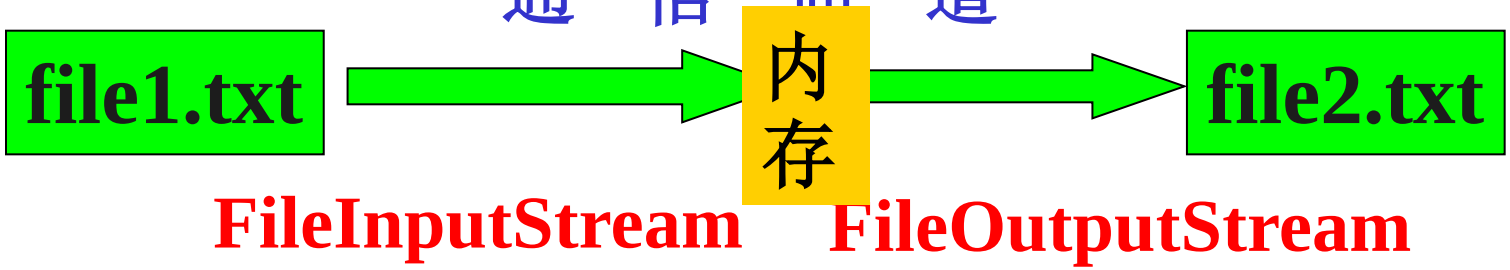


FileInputStream　　**FileOutputStream**

- 举例:利用程序,将文件**file1.txt**
- 拷贝到**file2.txt**中。
- 注意:要捕获文件异常

- 怎样实现这个程序呢?





- 文件类——**File**类（io包）

- 作用：**File**类主要用于命名文件、查询文件属性和处理文件目录。

- **public class File extends Object**
- **implements Serializable,Comparable**
- **{}**

- 构造函数
- File (String pathname)
- File (String parent , String child)
- File (File parent , String child)

例:File f1=new File("FileTest1.txt");

- //创建文件对象f1，f1所指的文件是在当前目录下创建的FileTest1.txt

例:File f2=new File("D:\\dir1","FileTest2.txt") ;

- // 注意：D:\\dir1目录事先必须存在，否则异常

例:File f4=new File("\\dir3");

File f5=new File(f4,"FileTest5.txt");

- //在如果 \\dir3目录不存在使用f4.mkdir()先创建

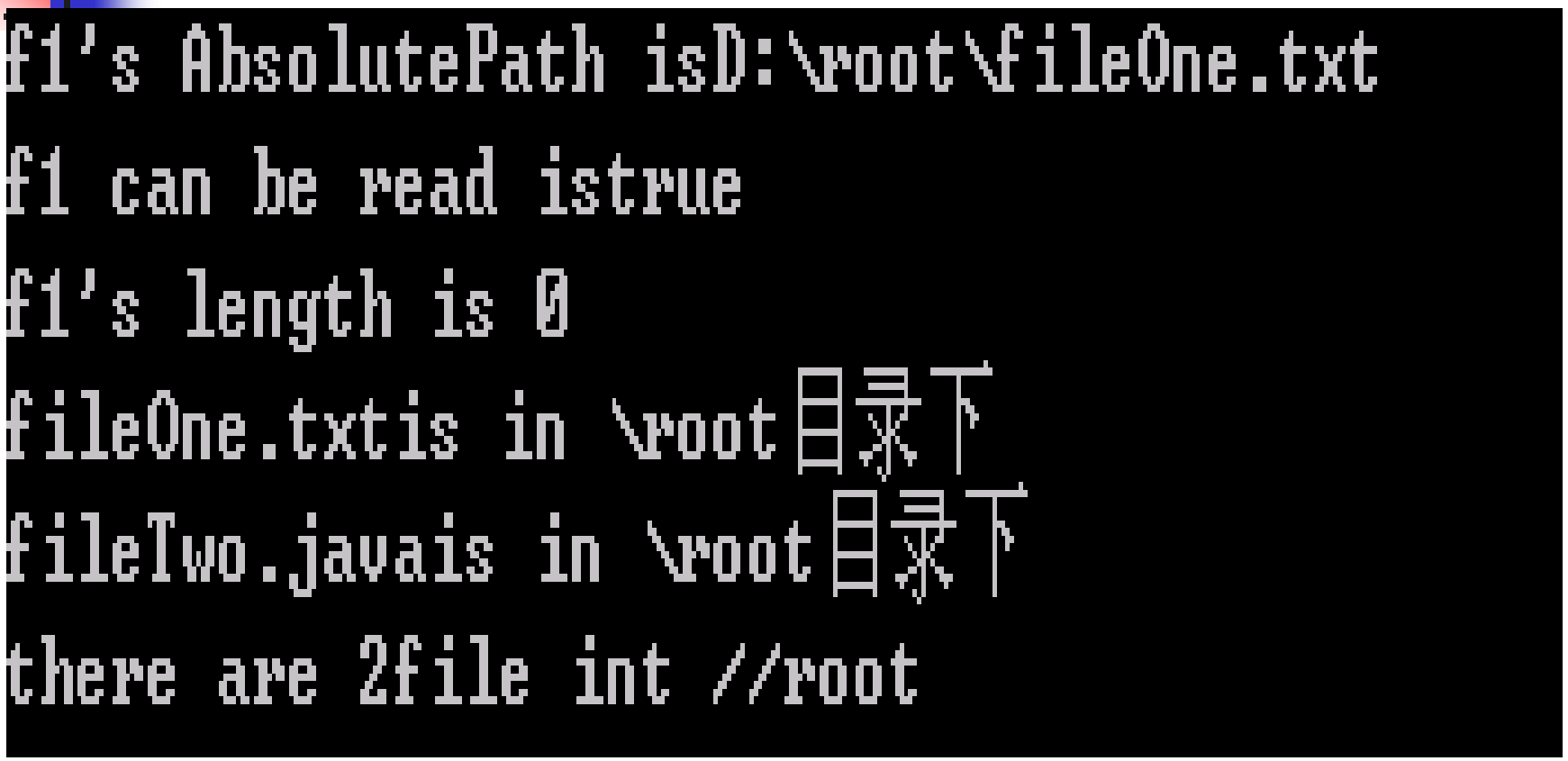
```
public class FileTestExp{
    public static void main(String args[])
                                    throws IOException {
    File dir=new File("\\root");
    File f1=new File(dir,  "fileOne.txt");
    File f2=new File(dir,  "fileTwo.java");
     // 文件对象创建后，指定的文件或目录不一定物理上存在
    if (!dir.exists())    dir.mkdir();
    if (!f1.exists())     f1.createNewFile();
    if (!f2.exists())    f2.createNewFile();
```

说明：File类的方法:(1) exists()测试磁盘中指定的文件或目录是否存在(2)mkdir()创建文件对象指定的目录（单层目录）(3)createNewFile()创建文件对象指定的文件

```
System.out.println("f1's AbsolutePath=
                    " +f1.getAbsolutePath());
System.out.println("f1 Canread="+f1.canRead());
System.out.println("f1's len= "+f1.length());
String []FL;  int count=0;      FL=dir.list( );
  for(int i=0;i<FL.length;i++){    count++;
       System.out.println(  FL[i]+"is in \\root");}
  System.out.println("there are"+count+
                                  "file in //root");   }
}
```

说明：File类的方法:

(4)list()返回目录中所有文件名字符串

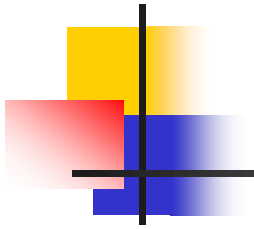

f1's AbsolutePath isD:\root\fileOne.txt
f1 can be read istrue
f1's length is 0
fileOne.txtis in \root目录下
fileTwo.javais in \root目录下
there are 2file int //root

返回

文件输入流： FileInputStream类

FileInputStream

- 作用：以文件作为数据输入源的数据流。
- 或者说是打开文件，从文件读数据到内存的类。



输入源的文件名

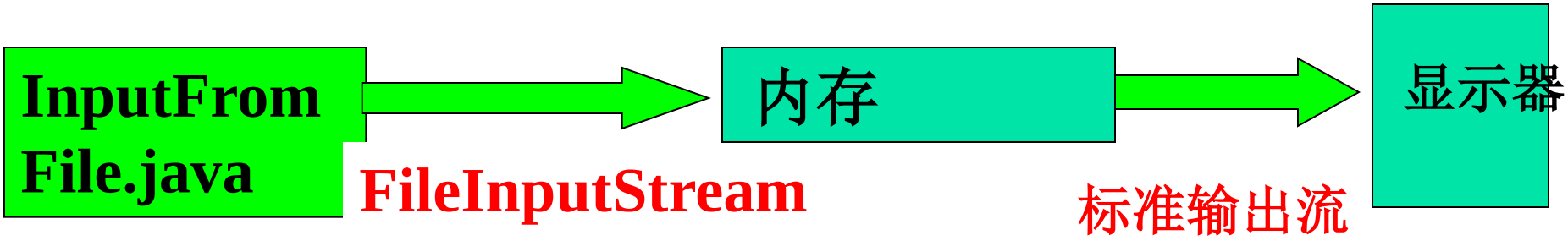
- 使用方法(1)

```
File  fin=new File("d:/abc.txt");
FileInputStream in=new FileInputStream( fin);
```

使用方法(2)

```
FileInputStream  in=new
                FileInputStream("d: /abc.txt");
```

- **程序举例：**
- 将**InputFromFile.java**的程序的内容显示在显示器上

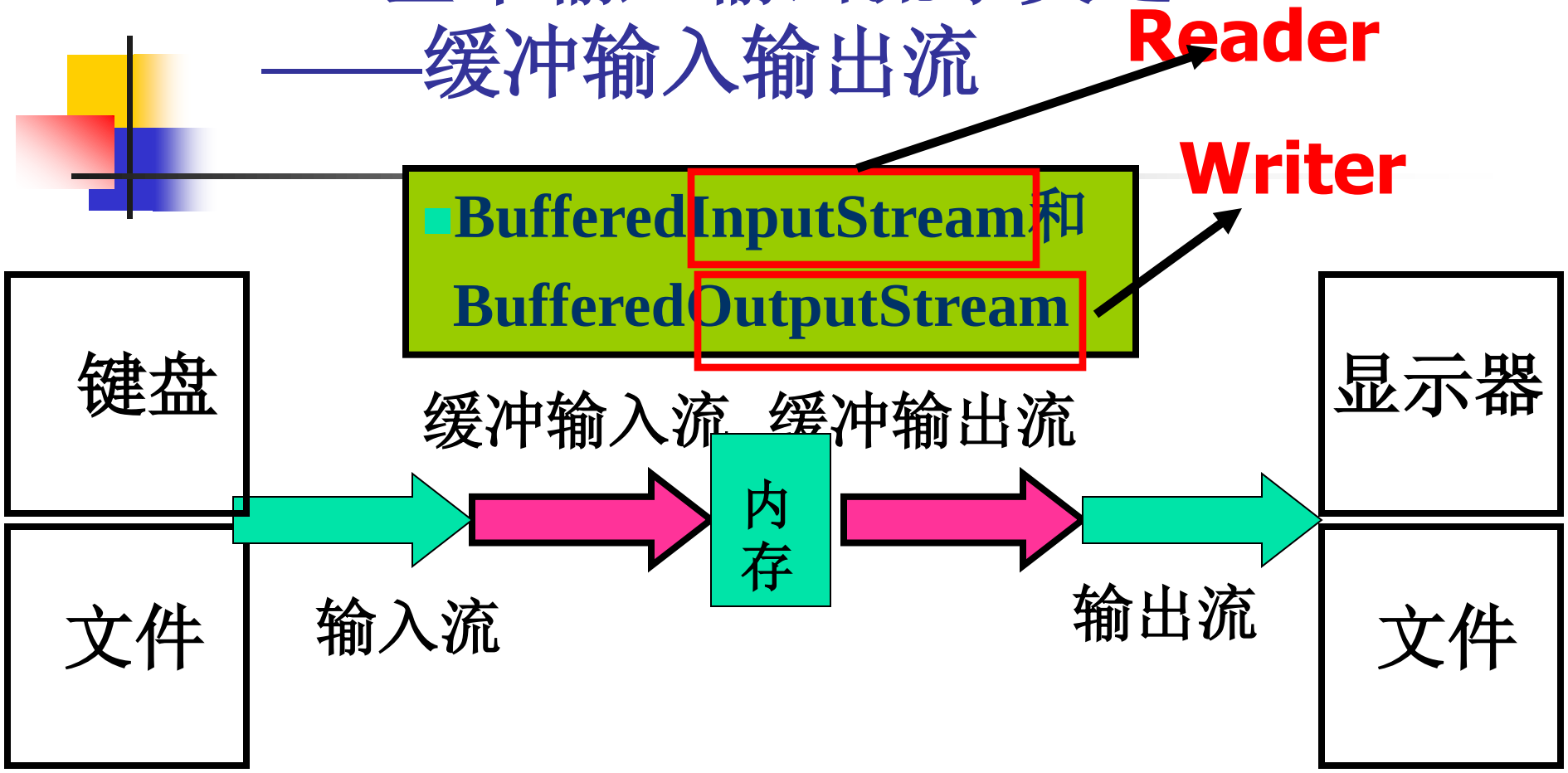


```
try{
      FileInputStream rf=new
                  FileInputStream("InputFromFile.java");
      int n=512;   byte  buffer[]=new  byte[n];
      while((rf.read(buffer,0,n)!=-1)&&(n>0)){
            System.out.println(new String(buffer) );
       }
      System.out.println();
      rf.close();
} catch(IOException  IOe){
     System.out.println(IOe.toString());
}
```

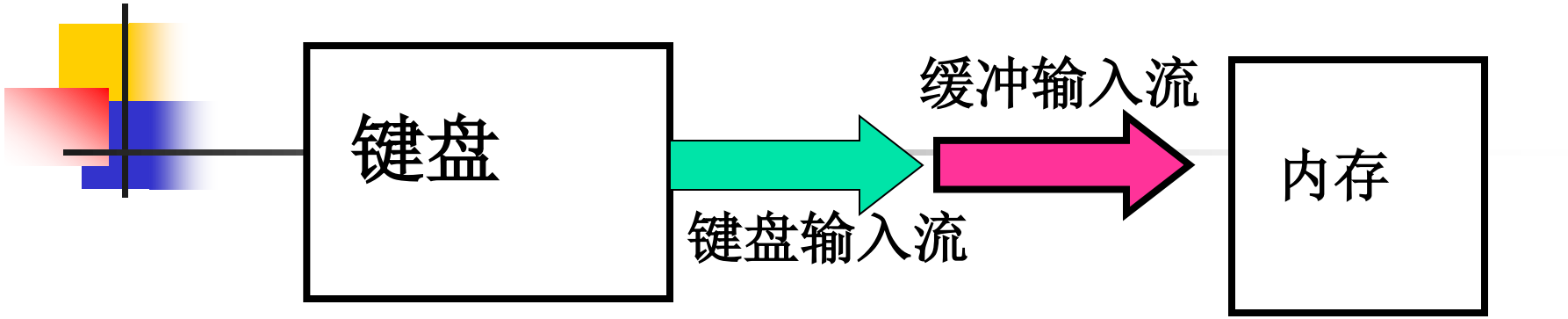
```
import java.io.*;
class InputFromFile{
        public static void main(String args[]){
                try{
          FileInputStream rf=new FileInputStream("InputFromFile.java");
              int n=512;   byte  buffer[]=new  byte[n];
          while((rf.read(buffer,0,n)!=-1)&&(n>0)){
            System.out.println(new String(buffer) );
          }
          System.out.println();
          rf.close();
        } catch(IOException  IOe){
         System.out.println(IOe.toString());
        }

        }
}
```

返回

（4）基本输入输出流子类之二
——缓冲输入输出流



- 目的：增加缓冲区流,减少访问硬盘的次数,提高效率



- 将BufferedReader与标准的数据流相接

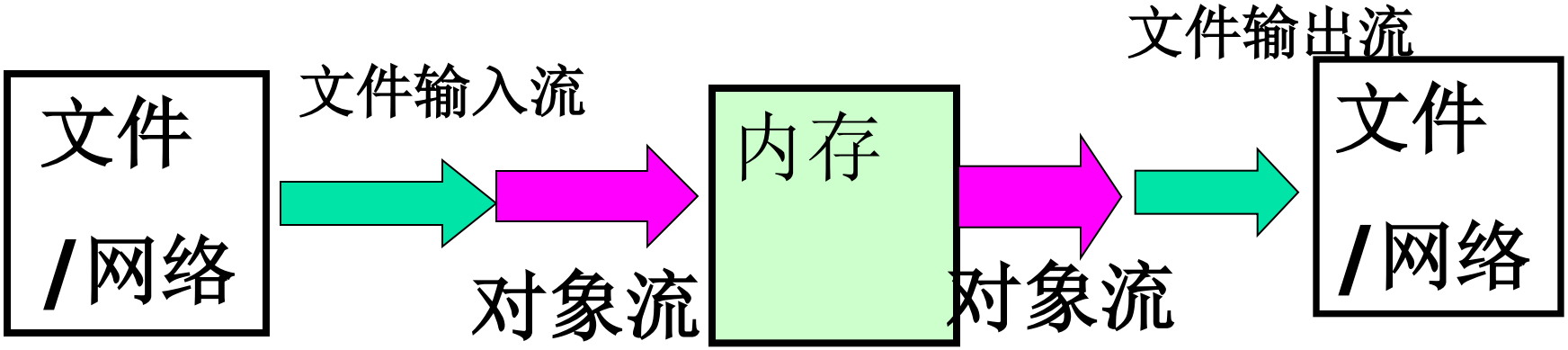
```
InputStreamReader sin= new
        InputStreamReader (System.in)
BufferedReader bin= new
                    BufferedReader(sin);
```

```
public class EncodeTest {
   private static void readBuff(byte [] buff) throws IOException {
      ByteArrayInputStream in =new ByteArrayInputStream(buff);
       int data;
       while((data=in.read())!=-1)   System.out.print(data+"  ");
       System.out.println();     in.close();     }
   public static voi
      System.out.pr
      char   c='好';
      int lowBit=c&
      System.out.pr
      String s="好"
      System.out.pr
      readBuff(s.get
      System.out.pr
      readBuff(s.get
      System.out.pr
      readBuff(s.getBytes("UTF-8"));     }
}
```

- 内存中采用**unicode**字符编码：
- **125 89**
- 本地操作系统默认字符编码：
- **186 195**
- 采用**GBK**字符编码：
- **186 195**
- 采用**UTF-8**字符编码：
- **229 165 189**

（6）对象流

- 定义：**能够输入输出对象的流称为对象流**
- （将对象在文件之间或网上进行传递）



ObjectInputStream　ObjectOutputStream

——对象序列化和对象流

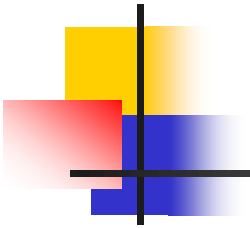
- （1）为什么序列化？
  - 对象不能够直接进行输入输出
- （2）什么是序列化？
  - 对象通过写出描述自己状态的的数值来记录自己的过程.
- （3）两者什么关系？
  - 将序列化后的对象通过对象输入输出流写入文件或传送到网络其他位置.

1.只有实现serializable接口的类才能被串行化

```
class Student implements Serializable
{ int id;      String name;
  int age;   String department;
  public Student(int id, String name,int age,
                              String department){
    this.id=id;  this.name=name;
    this.age=age;
    this.department =department;
  }
}
```

对象流举例:将Student类的一个实例写到文件中

```
import java.io.*;
public class Objectser{
  public static void main(String args[]) {
    Student stu=new
                Student(981036,"Li Ming",16,"CSD");
     try {
        FileOutputStream fo=new
                        FileOutputStream("date.ser");
       ObjectOutputStream so=new
                              ObjectOutputStream(fo);
       so.writeObject(stu); so.close();
     }catch(Exception e) {     }
  } }
```



对象流:从文件date.ser中读一个Student类的一个实例出来

```
Student stu=null;
try {
   FileInputStream fi=new
                        FileInputStream("date.ser");
   ObjectInputStream   si=new
                         ObjectInputStream(fi);
    stu=(Student) si.readObject();
    si.close();
 }catch(Exception e) {
    System.out.println(e.toString());   }
 System.out.println("ID: "+stu.id+"name:"+
   stu.name+"age:"+stu.age+"dept.:"+stu.departmen
```