Java IO1

- What?Why?How?

本章概述

- File类
- IO流的原理及概念
- IO流的分类
- IO流类的体系
- 字节流和字符流
- 处理流和节点流
- 文件拷贝

文件

什么是文件？

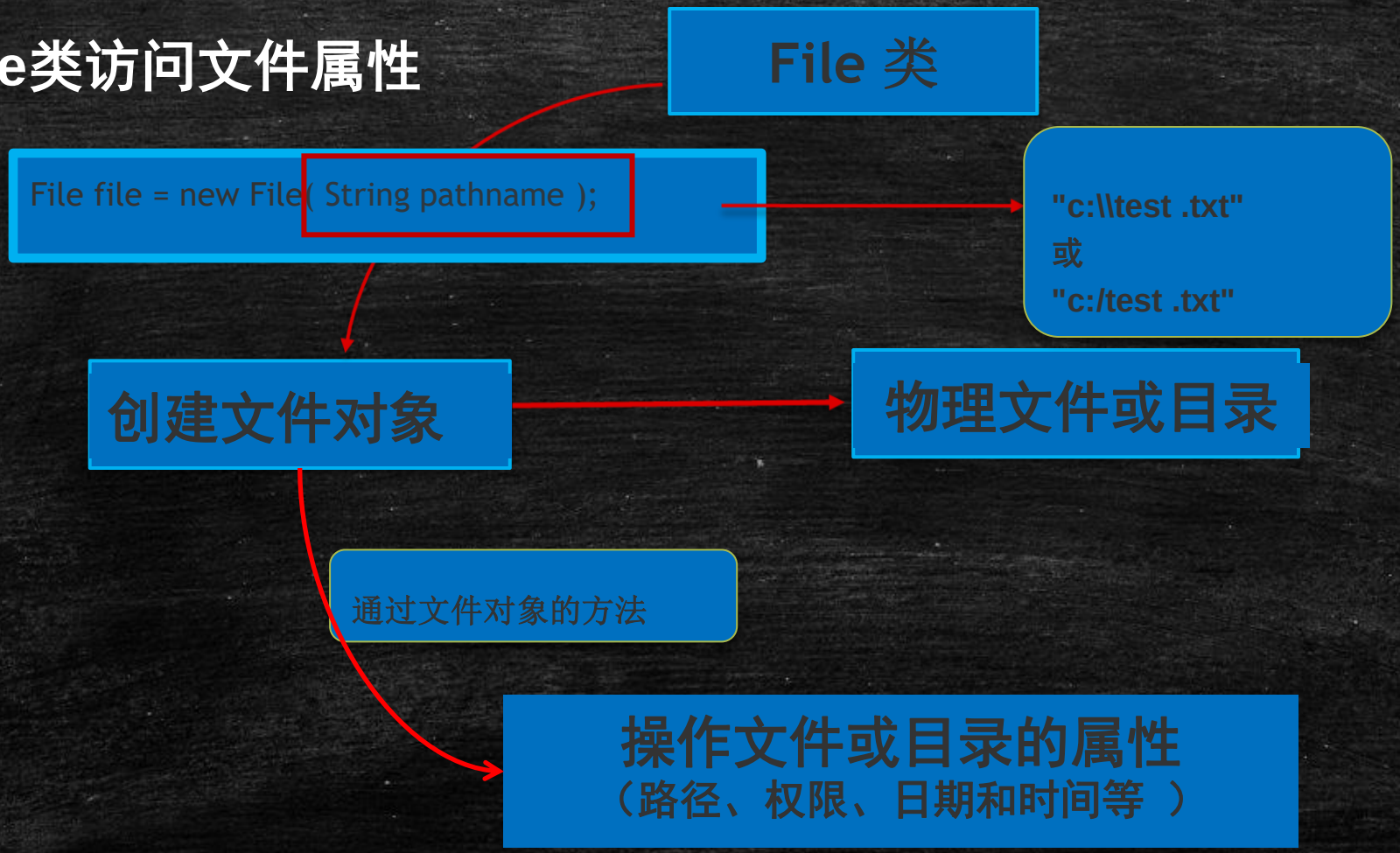
文件可认为是相关记录或放在一起的数据的集合

文件一般存储在哪里？

JAVA程序如何访问文件属性？

文件

- File类访问文件属性



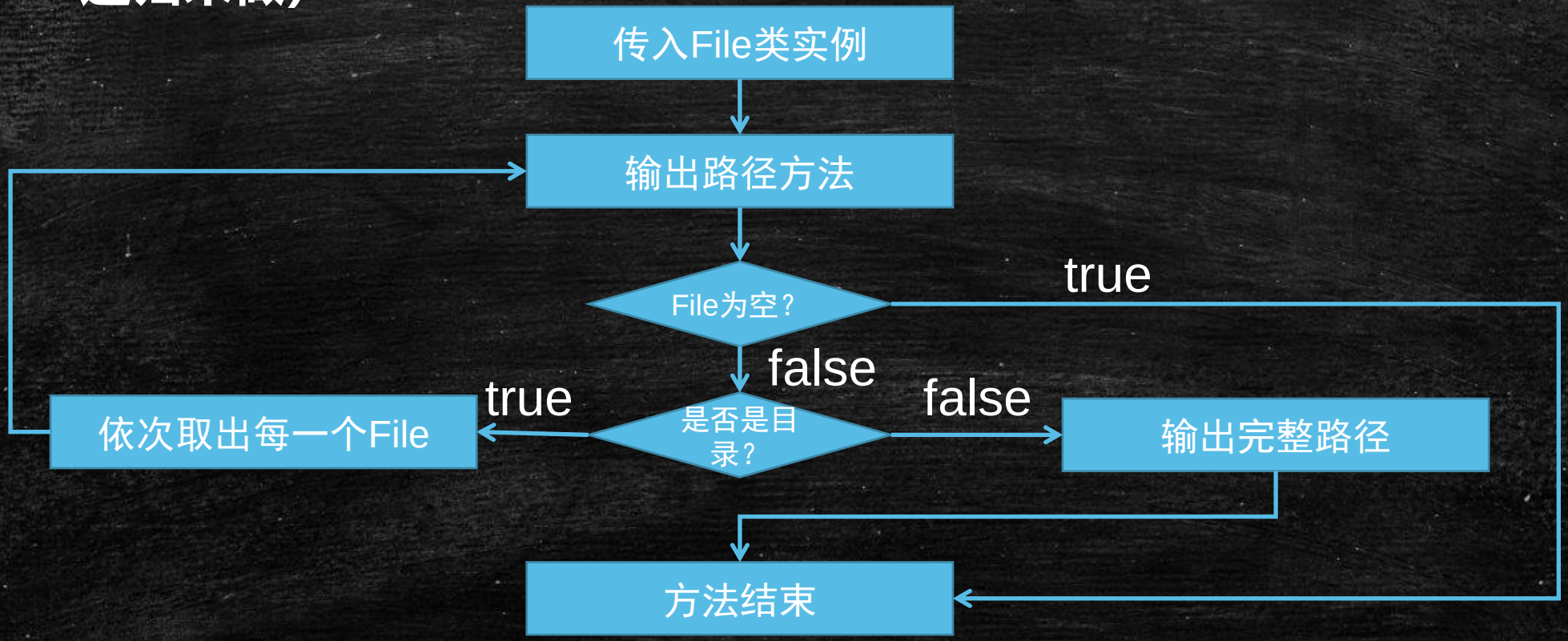
文件

- File类的常用方法

方法名称	说　　明
boolean exists( )	判断文件或目录是否存在
boolean isFile( )	判断是否是文件
boolean isDirectory( )	判断是否是目录
String getPath( )	返回此对象表示的文件的相对路径名
String getAbsolutePath( )	返回此对象表示的文件的绝对路径名
String getName( )	返回此对象表示的文件或目录的名称
boolean delete( )	删除此对象指定的文件或目录
boolean createNewFile( )	创建名称的空文件，不创建文件夹
long length()	返回文件的长度，单位为字节, 如果文件不存在，则返回 0L

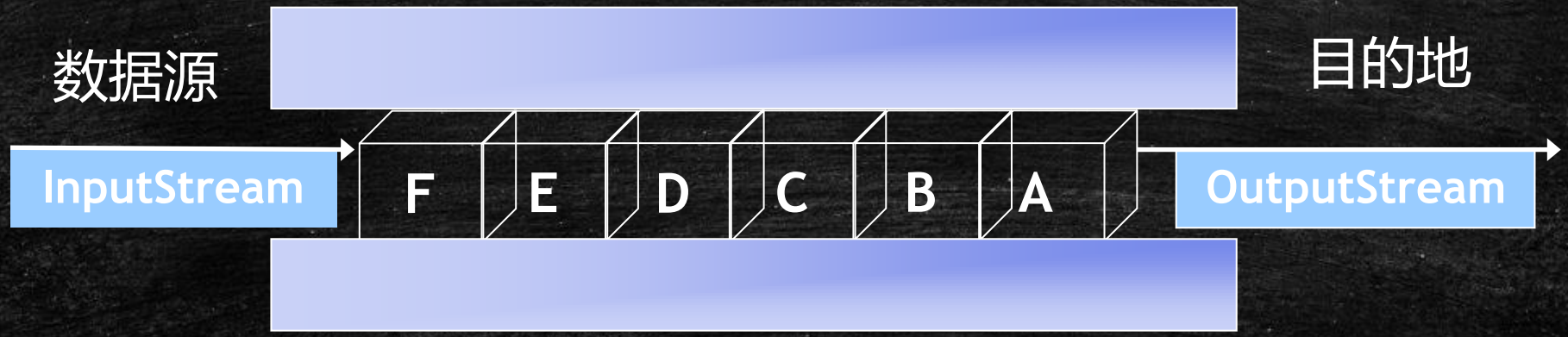
上机练习

- **编写一个程序，实现展现特定的文件夹及其子文件(夹)。（使用递归来做）**



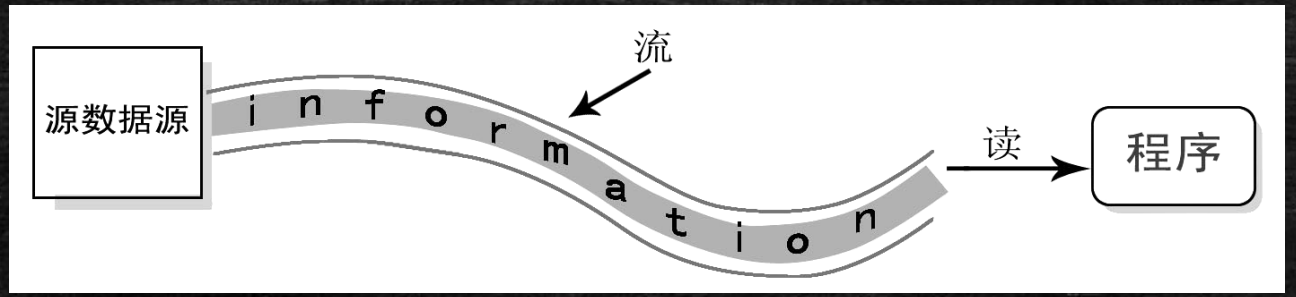
流的基本概念

- 如何读写文件?
- 通过流来读写文件
  - 流是指一连串流动的字符,是以先进先出方式发
- 送信息的通道

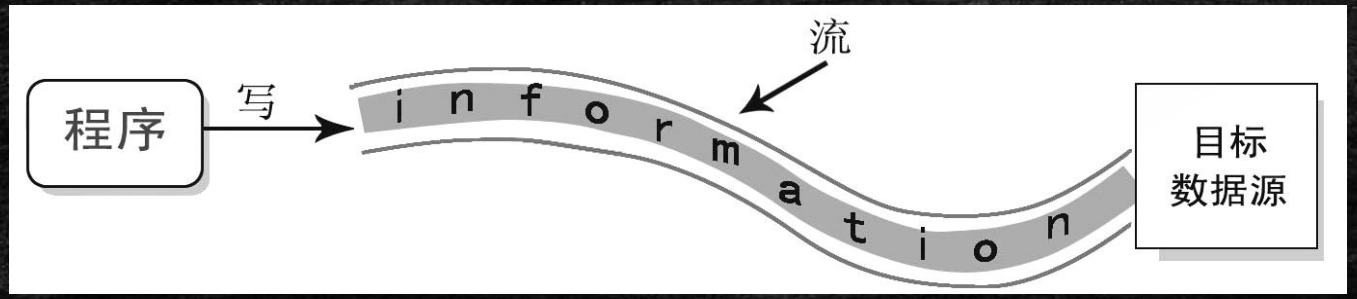


流的基本概念

- 输入/输出流与数据源
- XXX-→程序--→输入流



- 程序-→XXX--→输出流

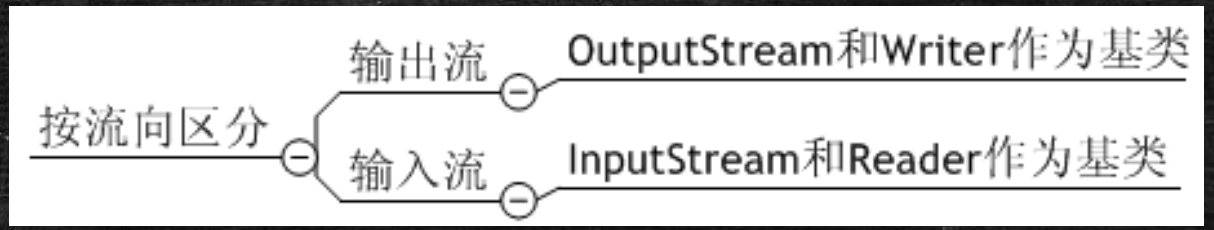


流的基本概念

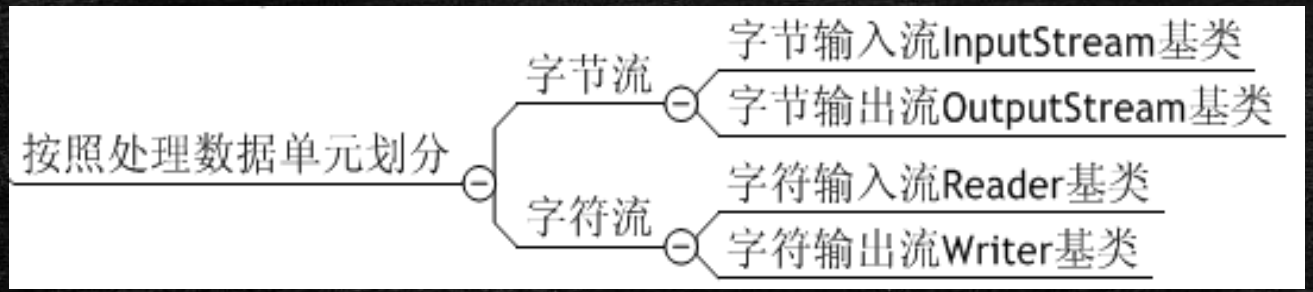
- **数据源**
  - data source. **提供原始数据**的
  - 原始媒介。常见的：数据库、
  - 文件、其他程序、内存、
  - 网络连接、IO设备。
  - **数据源就像水箱，流就像水管**
  - **中流着的水流，程序就是我们**
  - **最终的用户。** 流是一个抽象、
  - 动态的概念，是一连串连续动态
  - 的数据集合。

Java流的分类

- Java流的分类



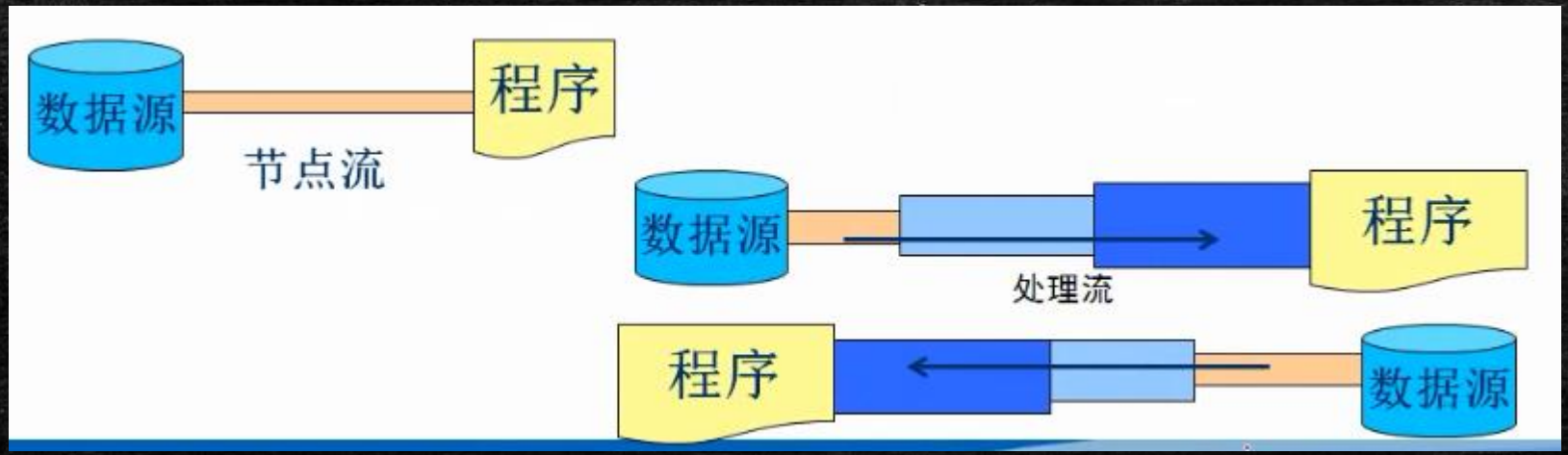
输入输出流是相对于计算机内存来说的,而不是相对于源和目标



字节流是 8 位通用字节流，字符流是 16 位 Unicode 字符流

流的分类

- 功能不同
  - 节点流：可以直接从数据源或目的地读写数据。
  - 处理流（包装流）：不直接连接到数据源或目的地，是其他流进行封装。目的主要是简化操作和提高性能。
- 节点流和处理流的关系
  - 节点流处于io操作的第一线，所有操作必须通过他们进行
  - 处理流可以对其他流进行处理（提高效率或操作灵活性）



文件的读写

- 文本文件的读写
  - 用FileInputStream和FileOutputStream读写文本文件
  - 用BufferedReader和BufferedWriter读写文本文件

- 二进制文件的读写
  - 使用DataInputStream和DataOutputStream读写二进制文件以及基本数据类型数据的读写

对象的读写

- 使用ObjectInputStream和ObjectOutputStream读写对象(序列化与反序列化)

使用FileInputStream 读文本文件

引入相关的类

构造文件输入流FileInputStream对象

读取文本文件的数据

关闭文件流对象

```
import java.io.IOException;
import java.io.FileInputStream;
```

```
FileInputStream fis= new
    FileInputStream("c:\\test.txt");
```

```
Int len=0;
While((len=fis.read()!=-1){
  .....
}
```

```
fis.close();
```

小结

- InputStream类常用方法
  - int read( )
  - int read(byte[] b)
  - int read(byte[] b,int off,int len)
  - void close( )
  - int available()

子类FileInputStream常用的构造方法

- FileInputStream(File file)
- FileInputStream(String name)

使用FileOutputStream 写文本文件

- 使用FileOutputStream 写文本文件的步骤与读文件的步骤有何不同？

引入相关的类

构造文件数据流输出对象

将数据写入文本文件

关闭文件流对象

```
import java.io.IOException;
import java.io.FileOutputStream;
```

```
FileOutputStrea fos = new
FileOutputStream        ("c:\\test.txt");
```

```
String str =“HelloWorld";
byte[] words  = str.getBytes();
fos.write(words, 0, words.length);
```

```
fos.close();
```

小结

- OutputStream类常用方法
  - void write(int c)
  - void write(byte[] buf)
  - void write(byte[] b,int off,int len)
  - void close( )
- 子类FileOutputStream常用的构造方法
  - FileOutputStream (File file)
  - FileOutputStream(String name)
  - FileOutputStream(String name,boolean append)

1、前两种构造方法在向文件写数据时将覆盖文件中原有的内容
2、创建FileOutputStream实例时，如果相应的文件并不存在，则会自动创建一个空的文件

上机练习

- 训练要点
  - 理解输入流和输出流类的概念。
  - 使用FileInputStream实现读取文本文件。
  - 使用FileOutputStream实现向文本文件中写数据
- 需求说明
  - 文件“我的青春谁做主.txt”位于D盘根目录下，要求将此文件的内容复制到
  - C:\myFile\myPrime.txt中

上机练习

- **复制图片**
- 将D:\盘指定的图片复制到当前项目中
- **需求分析：**
- 使用字节的输入流FileInputStream读取字节
- 使用字节的输入流FileOutputStream写入文件

使用字符流读写文件文件

读文件　　使用Reader抽象类实现

写文件　　使用Writer抽象类实现

使用FileReader读取文件

```
import java.io.Reader;
import java.io.FileReader;
import java.io.IOException;
... ...
//创建 FileReader对象对象
Reader fr=null;
StringBuffer sbf=null;
try {
    fr = new FileReader("D:\\myDoc\\简介.txt");
    char ch[]=new char[1024];  //创建字符数组作为中转站
    sbf=new StringBuffer();
    int length=fr.read(ch);  //将字符读入数组
    while ((length!= -1)) {//循环读取并追加字符
        sbf.append(ch);   //追加到字符串
        length=fr.read();
    }
}... ...
    fr.close();
```

1、引入相关的类

与字节流FileInputStream类实现文本文件读取步骤类似

2、创建FileReader对象

3、读取文本文件的数据

4、关闭相关的流对象

BufferedReader类

如何提高字符流读取文本文件的效率？

- 使用FileReader类与BufferedReader类

BufferedReader类是Reader类的子类
BufferedReader类带有缓冲区
按行读取内容的readLine()方法

BufferedReader类特有的方法

使用 BufferedReader 读文本文件

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
```

```
Reader fr=new
            FileReader("C:\\myTest.txt ");
BufferedReader br=new
            BufferedReader(fr);
```

```
br.readLine();
```

```
br.close();
fr.close();
```

- 通过字符流的方式读取文件，并使用缓冲区，提高读文本文件的效率

小结

Reader类常用方法

- int read( )
- int read(byte[] c)
- read(char[] c,int off,int len)
- void close( )

子类BufferedReader常用的构造方法

- BufferedReader(Reader in)

子类BufferedReader特有的方法

- readLine()

使用FileWriter写文件

```
import java.io.Reader;
import java.io.FileWriter ;
import java.io.IOException;
… …
try {
    //创建一个FileWriter对象
    fw=new FileWriter("D:\\myDoc\\简介.txt");
    //写入信息
    fw.write("我热爱我的团队！ ");
    fw.flush();  //刷新缓冲区
}catch(IOException e){
    System.out.println("文件不存在!");
}
… …
    fw.close();  //关闭流
… …
```

1、引入相关的类

2、创建FileWriter对象

3、写文本文件

与字节流FileOutputStream类实现向文本文件写入数据步骤类似

4、关闭相关的流对象

BufferedWriter类

- 如何提高字符流写文本文件的效率？
  - 使用FileWriter类与BufferedWriter类

BufferedWriter类是Writer类的子类
BufferedWriter类带有缓冲区

使用 BufferedWriter 写文件

```
import java.io.FileWriter ;
import java.io.BufferedWriter ;
import java.io.IOException;
```

```
FileWriter fw = new
            FileWriter("C:\\myTest.txt");
BufferedWriter bw=new
            BufferedWriter(fw);
```

```
bw.write("hello");
```

```
bw.flush();
fw.close();
```

小结

Writer类常用方法

- write(String str)
- write(String str,int off,int len)
- void close()
- void flush()

子类BufferedWriter常用的构造方法

- BufferedReader(Writer out)

上机练习

【1】使用File类的方法去创建一个文本文件，先进行判断

如果没有则创建，如有有则先删除再创建

【2】使用BufferedWriter将如下文字

《虞美人》

春花秋月何时了?

往事知多少。

小楼昨夜又东风，

故国不堪回首月明中。

雕栏玉砌应犹在，

只是朱颜改。

问君能有几多愁?

恰似一江春水向东流。

写入【1】中所创建的文件

【3】再将【2】中写入的文件读取到控制台输出

上机练习

- 【1】创建User类，包含以下属性name:String,age:int
- ,gender enum,重写toString方法显示对象的信息
- 【2】使用BufferedWriter写入文件以“,”分隔
- 【3】使用BufferedReader读取信息并进行分割，还原成对象，调用
- 对象的toString方法输出对象的信息

本章总结

- File 类用于访问文件或目录的属性
- 程序和数据源之间通过流联系
  - 输入流和输出流
  - 字节流和字符流
  - 节点流和包装流
- FileInputStream和FileOutputStream以字节流的方式读写文本文件。
- BufferedReader和BufferedWriter以字符流的方式读写文本文件，而且效率更高。