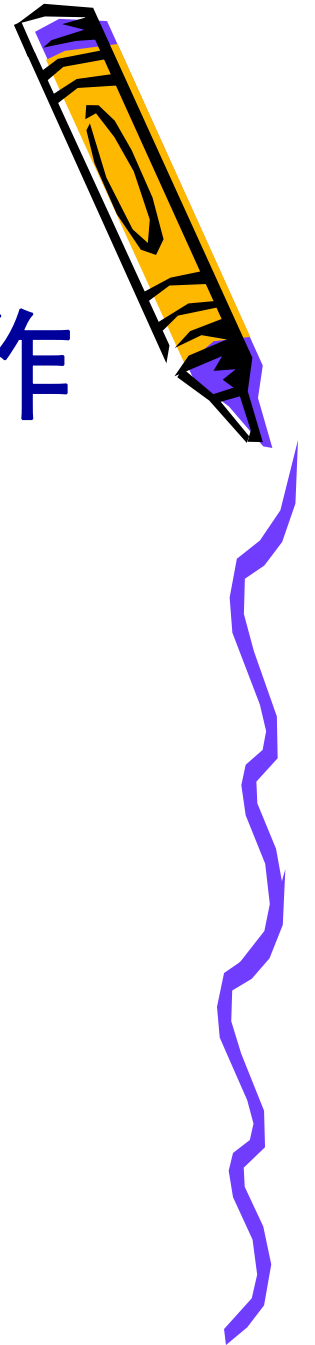


# 第8章 输入/输出流和文件操作



- 8.1 文件和流的概念
- 8.2 字节输入/输出流类
- 8.3 字符输入/输出流类
- 8.4 文件操作



# 第8章 输入/输出流和文件操作

## 内容和要求:

1. 理解流的概念; 掌握字节流和字符流对类型文件和文本文件进行顺序处理; 熟悉在对象之间通过流传递数据的方法; 了解**Java**的标准输入/输出方法。
2. 掌握文件操作, 掌握**File**类, 熟悉文件过滤器、文件对话框; 了解**RandomAccessFile**随机存取文件类。

**重点:** 各种字节流类, 字符流类, **File**类。

**难点:** ① 如何选择使用哪种字节流或字符流, 掌握程序设计方法, 而不是死记硬背。

② 操作系统的文件组织方式是树结构, 递归算法。





## 8.1 文件和流的概念

### 8.1.1 操作系统中的文件和目录概念

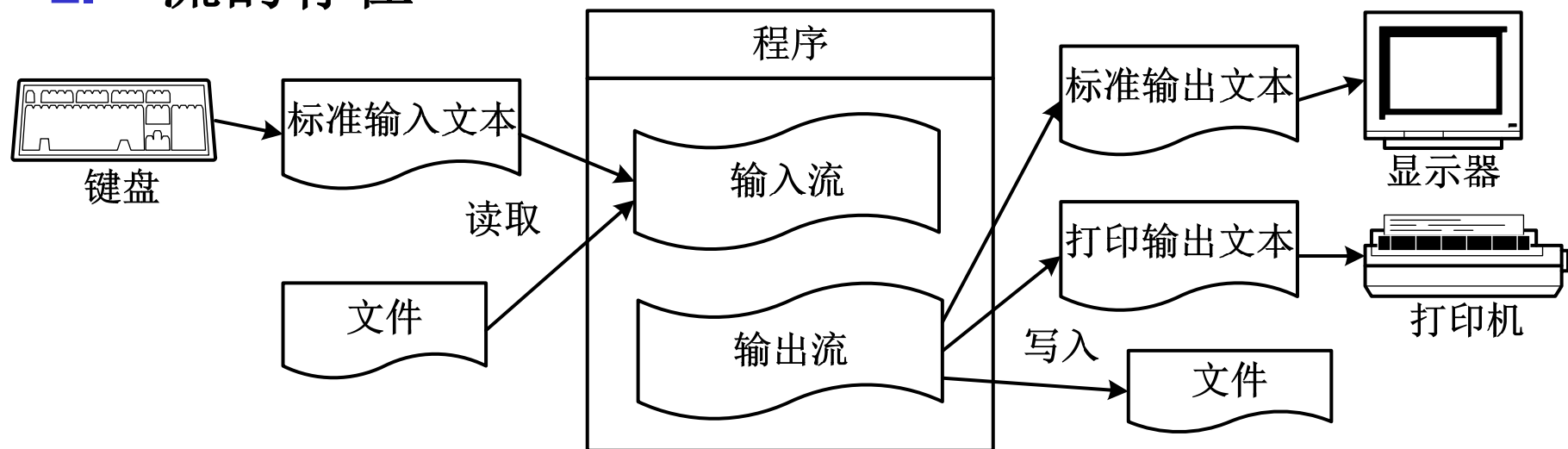
1. 文件与文件系统
2. 目录结构与文件检索
3. 文件的逻辑结构
  - ① 流式文件
  - ② 记录式文件
4. 文件的存取方法
  - ① 顺序存取
  - ② 随机存取
5. 文件的使用
  - ① 操作接口
  - ② 应用程序接口

## 8.1.2 流的概念

### 1. 流的定义和作用

- ① 流的定义、方向性和读/写操作
- ② 流采用缓冲区技术
- ③ 流的作用

### 2. 流的存在





## 3. Java的流类与文件类

---

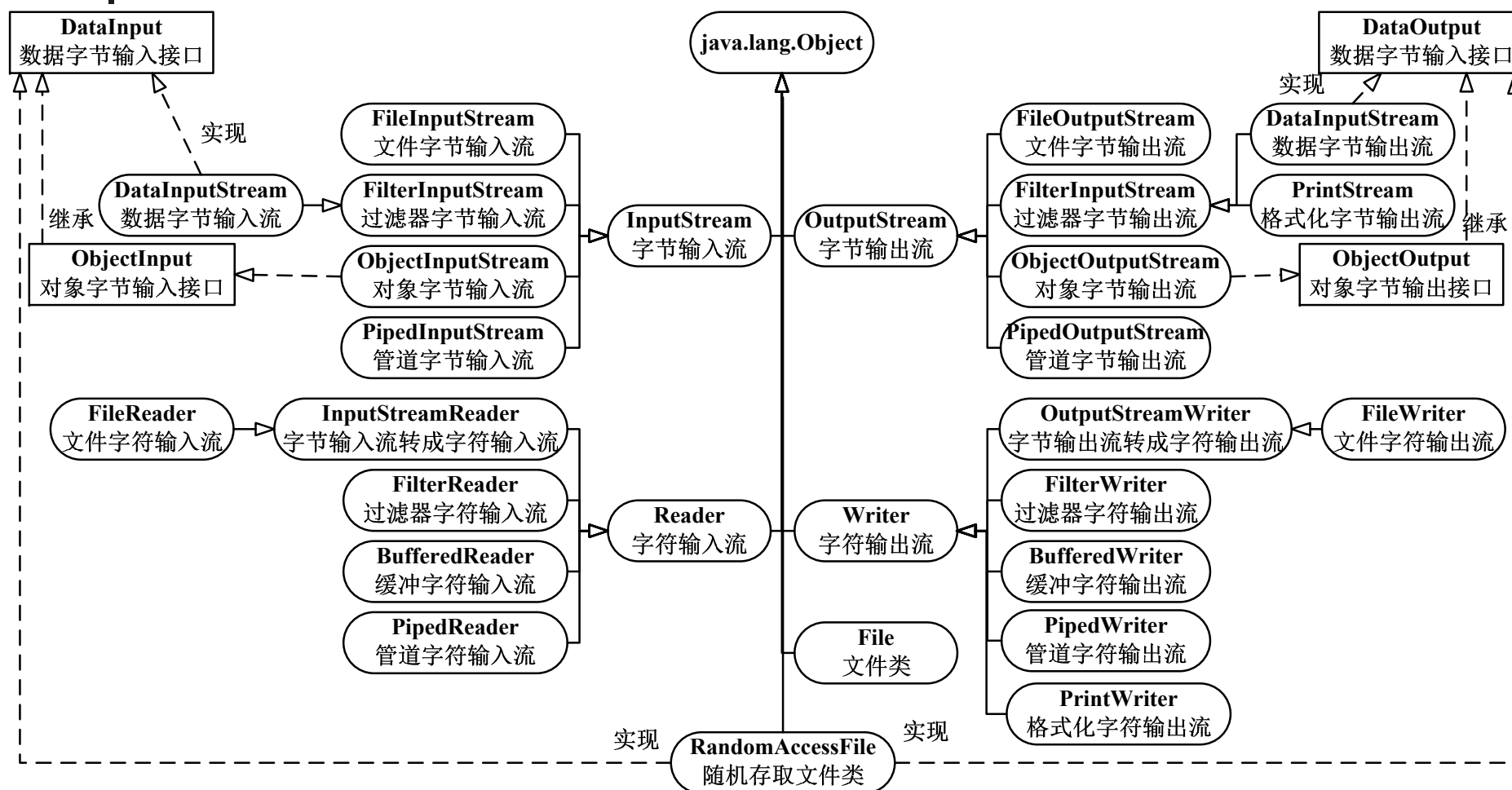
### 1. 流类

- |                       |          |
|-----------------------|----------|
| ① <b>InputStream</b>  | 抽象字节输入流类 |
| ② <b>OutputStream</b> | 抽象字节输出流类 |
| ③ <b>Reader</b>       | 抽象字符输入流类 |
| ④ <b>Writer</b>       | 抽象字符输出流类 |

### 2. 文件类

- |                           |         |
|---------------------------|---------|
| ① <b>File</b>             | 文件类     |
| ② <b>RandomAccessFile</b> | 随机存取文件类 |

# 3. Java的流类与文件类





## 8.2 字节输入/输出流类

---

- 8.2.1 抽象字节流
- 8.2.2 文件字节流
- 8.2.3 数据字节流
- 8.2.4 对象字节流
- 8.2.5 管道字节流

## 8.2.1 抽象字节流

### 1、InputStream抽象字节输入流类



```
public abstract class InputStream extends  
Object implements Closeable
```

```
{
```

```
    public abstract int read() throws IOException;
```

```
        //返回读取的一个字节，抽象方法
```

```
    public int read(byte[] buffer) throws IOException
```

```
        //从输入流中读取若干字节到指定缓冲区，返回实  
        际读取的字节数
```

```
    public void close() throws IOException //关闭流
```

```
}
```

【思考题8-1】① **read()**方法读取**1**个字节，返回值类型是**int**，而不是**byte**，为什么？

② **read()**方法为什么能够将**-1**作为输入流结束标记，**-1**不是**int**整数吗？

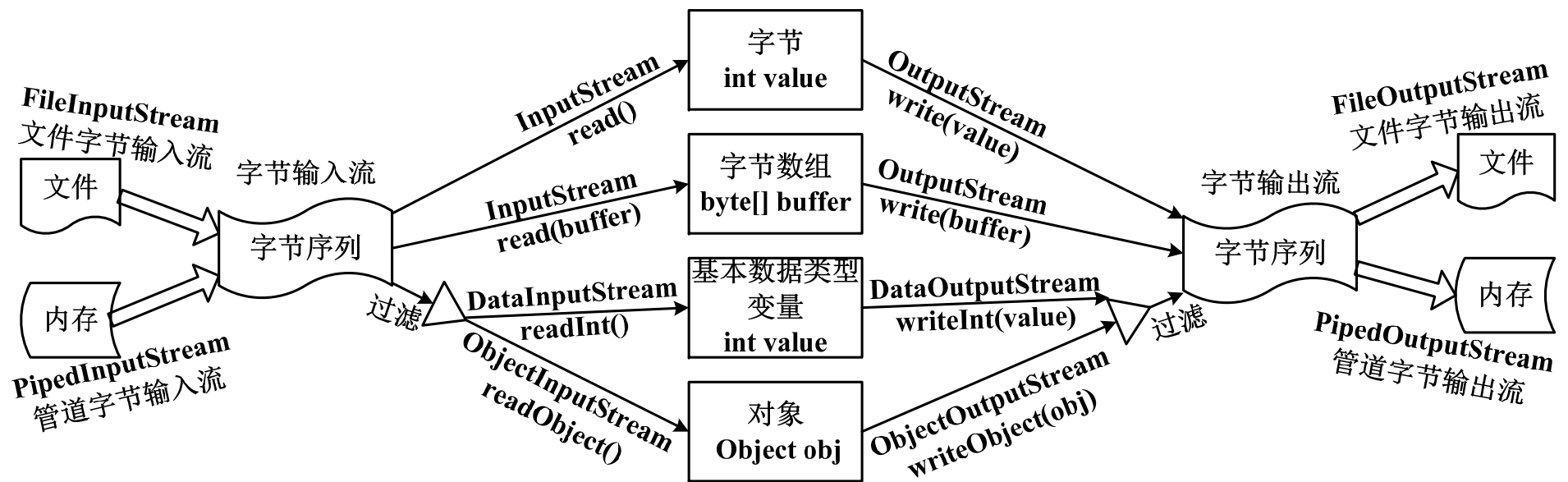


## 2. OutputStream抽象字节输出流类

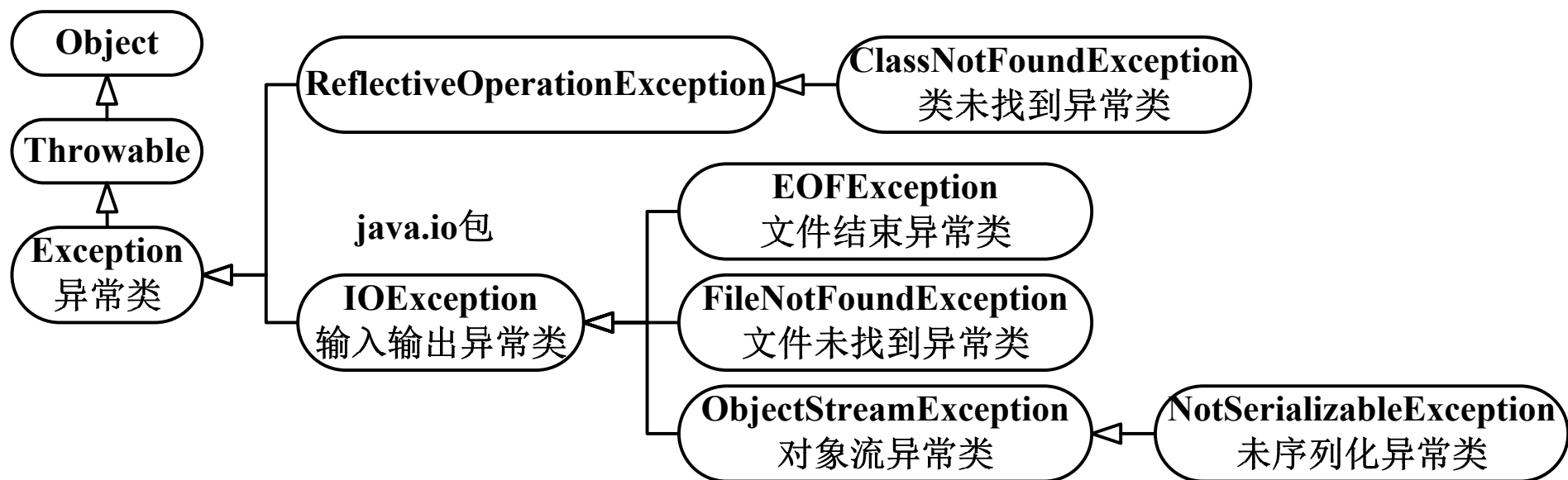
```
public abstract class OutputStream extends Object
    implements Closeable, Flushable
{
    public abstract void write(int b) throws
        IOException;           //写入1个字节，抽象方法
    public void write(byte[] buffer) throws
        IOException           //将字节数组写入字节流
    public void flush() throws IOException //立即传输
    public void close() throws IOException //关闭流
}
```

【思考题8-1】① **write(int)**方法写入1个字节，参数类型是**int**，而不是**byte**，为什么？

## 图8.2 各种字节输入/输出流的读/写方法



## 图8.3 使用流类和文件类出现异常类的层次结构





## 8.2.2 文件字节流

---

### 1. **FileInputSream**文件字节输入流类

```
public class FileInputStream extends  
    InputStream  
{  
    public FileInputStream(String filename)  
        throws FileNotFoundException  
    public FileInputStream(File file) throws  
        FileNotFoundException  
}
```

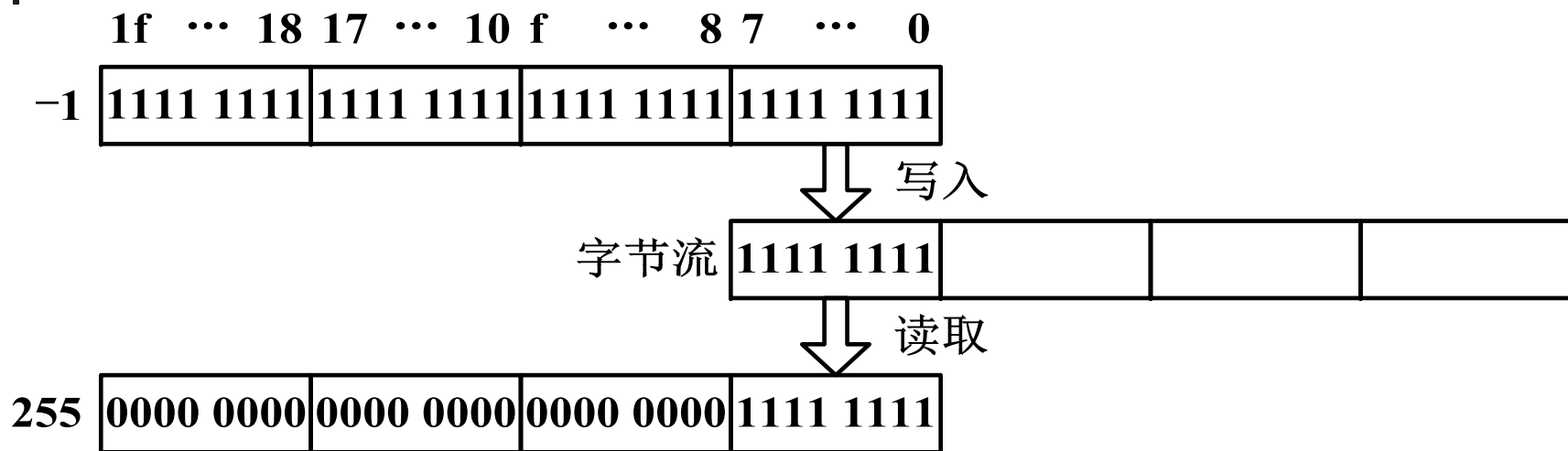


## 2. **FileOutputStream** 文件字节 输出流类

```
public class FileOutputStream extends  
    OutputStream  
{  
    public FileOutputStream(String filename)  
        throws FileNotFoundException  
    public FileOutputStream(File file) throws  
        FileNotFoundException  
    public FileOutputStream(String filename,  
        boolean append) throws  
        FileNotFoundException  
}
```

## 【例8.1】 理解字节流。

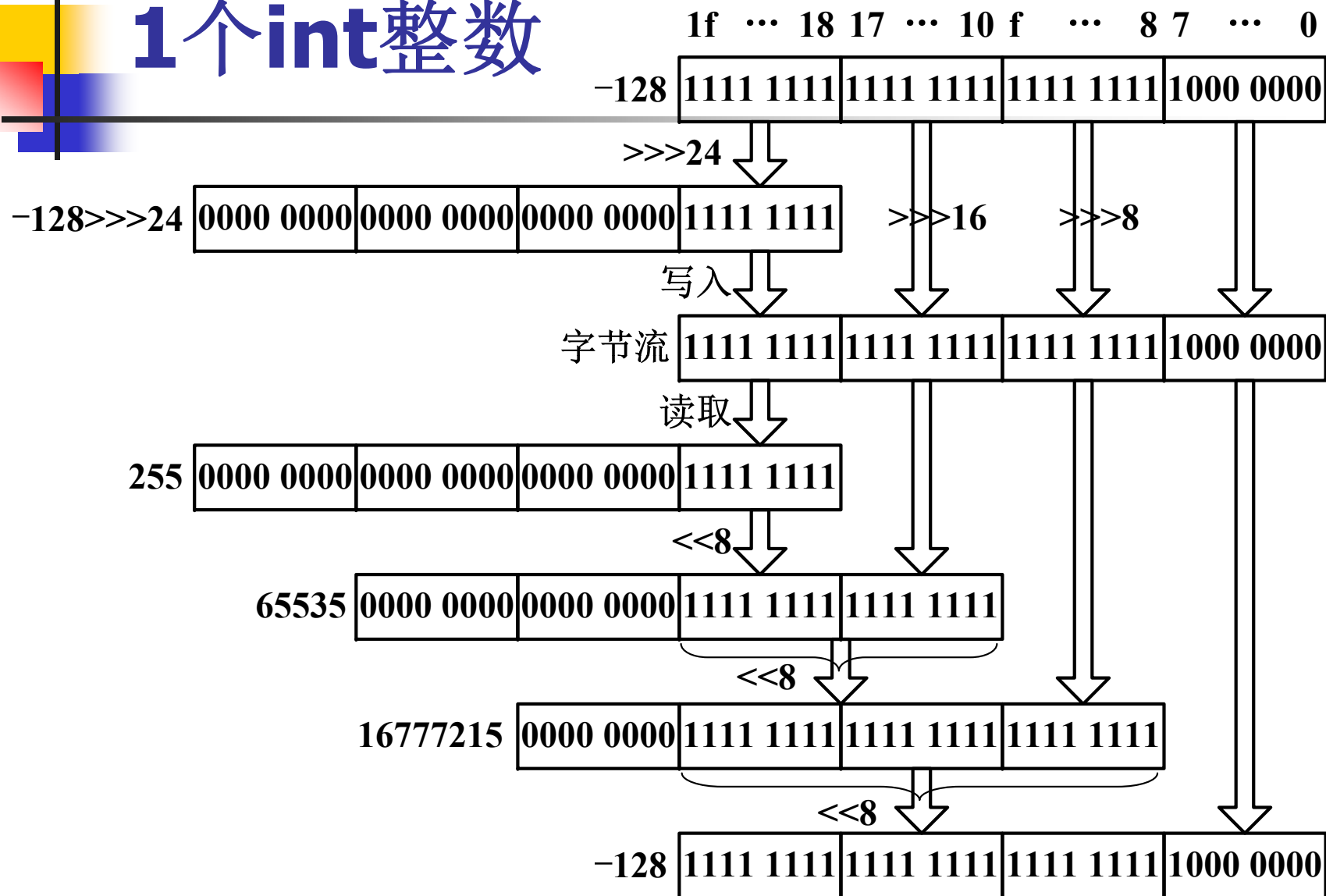
### (1) 从文件字节流中读写1个字节



【思考题8-2】 向字节流写入1字节-256和256，再各读取1字节，值为多少？为什么？

【答】 0，因为-256 (1.....1 00000000)；  
0， 256 (0.....01 00000000)

## (2) 从字节流中读写4个字节作为1个int整数





## (3) 文件复制操作

---

### 【思考题8-2】

1. 从字节流读写**2**个字节表示**short**整数或**char**字符等。
2. **public static boolean equals(String filename1, String filename2)**  
//比较两个文件内容是否相同






## 8.2.3 数据字节流

### 1. **DataInputStream**数据字节输入流类

```
public class DataInputStream extends  
    FilterInputStream implements DataInput  
{  
    public DataInputStream(InputStream in)           //构造方法  
    public final byte readByte() throws IOException  
    public final short readShort() throws IOException  
    public final int readInt() throws IOException    //读取整型  
    public final long readLong() throws IOException  
    public final float readFloat() throws IOException  
    public final double readDouble() throws IOException  
    public final char readChar() throws IOException //读取字符  
    public final boolean readBoolean() throws IOException  
}
```

## 2. DataOutputStream数据字节 输出流类

```
public class DataOutputStream extends FilterOutputStream
    implements DataOutput
{
    public DataOutputStream(OutputStream out)           //构造方法
    public final void writeByte(int v) throws IOException
    public final void writeShort(int v) throws IOException
    public final void writeInt(int v) throws IOException //写入整型
    public final void writeLong(long v) throws IOException
    public final void writeFloat(float v) throws IOException
    public final void writeDouble(double v) throws IOException
    public final void writeChar(int v) throws IOException //写入字符
    public final void writeBoolean(boolean v) throws IOException
    public final void writeChars(String s) throws IOException
}
```



## 【例8.2】 采用整数文件保存 随机数序列，使用数据流读写 整数。

### 【思考题8-3】

1. 增加求最大值、最小值和平均值等计算功能。
2. 将创建**FileOutputStream**流对象语句改为如下，  
程序运行结果是怎样的？

```
FileOutputStream fout = new  
FileOutputStream(filename,true);  
//文件字节输出流，添加方式
```



## 8.2.4 对象字节流

### 1. **ObjectInputStream**对象字节输入流类

```
public class ObjectInputStream extends  
    InputStream implements ObjectInput,  
    ObjectStreamConstants  
{  
    public ObjectInputStream(InputStream in)  
        throws IOException           //构造方法  
    public final Object readObject() throws  
        IOException, ClassNotFoundException  
        //读取一个对象  
}
```



## 2. ObjectOutputStream对象字节输出流类

```
public class ObjectOutputStream extends  
    OutputStream implements ObjectOutput,  
    ObjectStreamConstants  
{  
    public ObjectOutputStream(OutputStream  
        out) throws IOException //构造方法  
    public final void writeObject(Object obj)  
        throws IOException //写入一个对象  
}
```

## 【例8.3】 使用对象流读写对象文件。



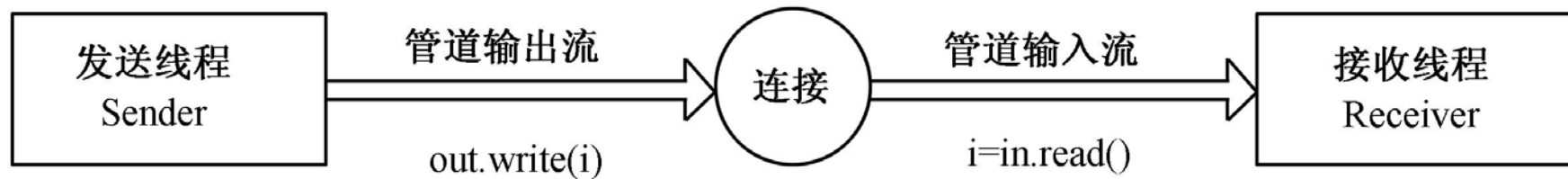
```
public class MyDate implements  
    Comparable<MyDate>, Serializable
```

```
public class Person implements  
    Comparable<Person>, Serializable
```

```
public class Student extends Person
```

```
public class FilePersonJFrame extends  
    PersonJFrame implements  
    WindowListener // 继承例6.4框架
```

## 8.2.5 管道字节流

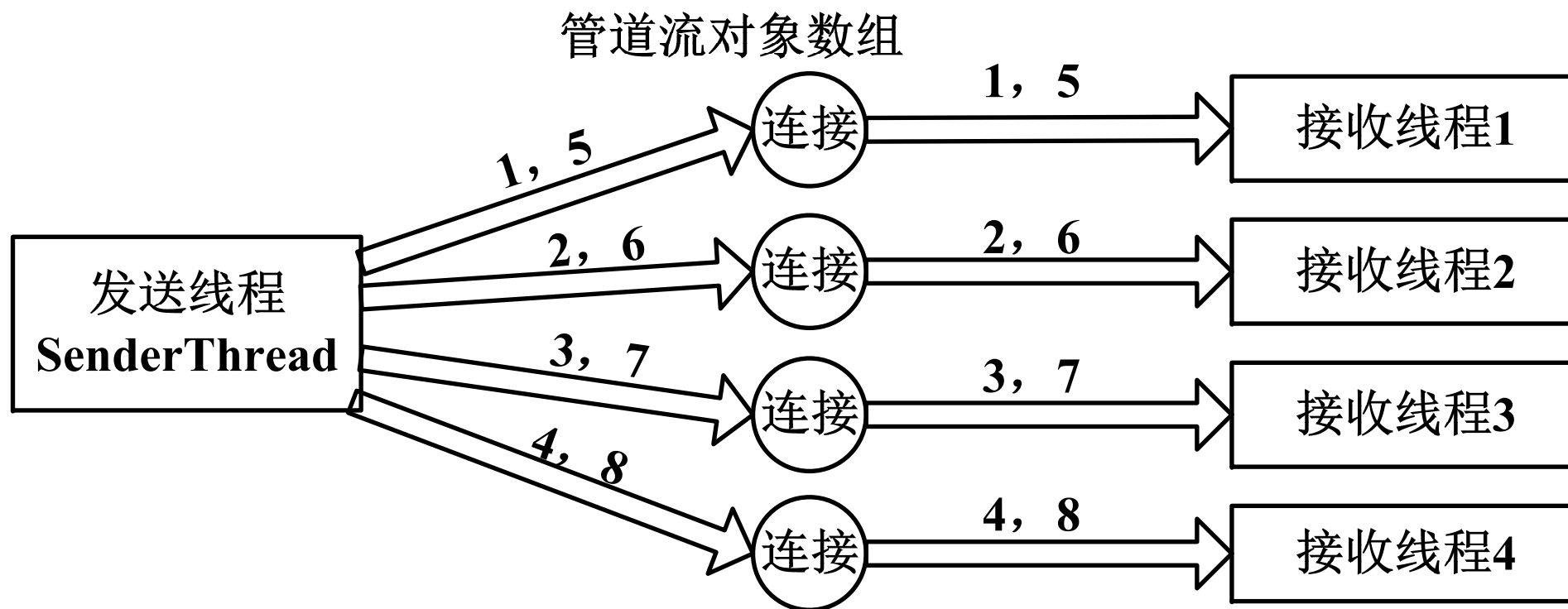


1. **PipedInputStream** 管道字节输入流类
2. **PipedOutputStream** 管道字节输出流类

```
PipedInputStream in = new PipedInputStream();  
try  
{  
    PipedOutputStream out= new  
        PipedOutputStream(in);  
}  
catch(IOException ioe) {}
```

## 【例8.4】 使用管道流实现的发牌程序。

图8.9 发牌程序中多个线程对象间的管道流







## 【思考题8-5】

**ReceiveCardJFrame类run()方法中的while(true) 循环何时结束？**

**【答】① SenderThread发牌线程类调用：**  
**douts[i].close() //关闭数据字节输出流**

**则ReceiveCardJFrame的din.readInt()方法捕获到EOFException异常（IOException的子类）。**

**② 如果SenderThread发牌线程类没有调用douts[i].close()方法，则ReceiveCardJFrame的din.readInt()捕获到IOException异常。**



## 8.3 字符输入/输出流类

---

- 8.3.1 抽象字符流
- 8.3.2 字节/字符转换流
- 8.3.3 文件字符流
- 8.3.4 缓冲字符流
- 8.3.5 格式化字符输出流
- 8.3.6 Java标准输入/输出



## 8.3.1 抽象字符流

---

### 1. **Reader**抽象字符输入流类

**public abstract class Reader extends Object**  
**implements Readable, Closeable**

**{**

**public int read() throws IOException**

**public int read(char cbuffer[]) throws**  
**IOException**

**public abstract void close() throws**  
**IOException;**

**}**

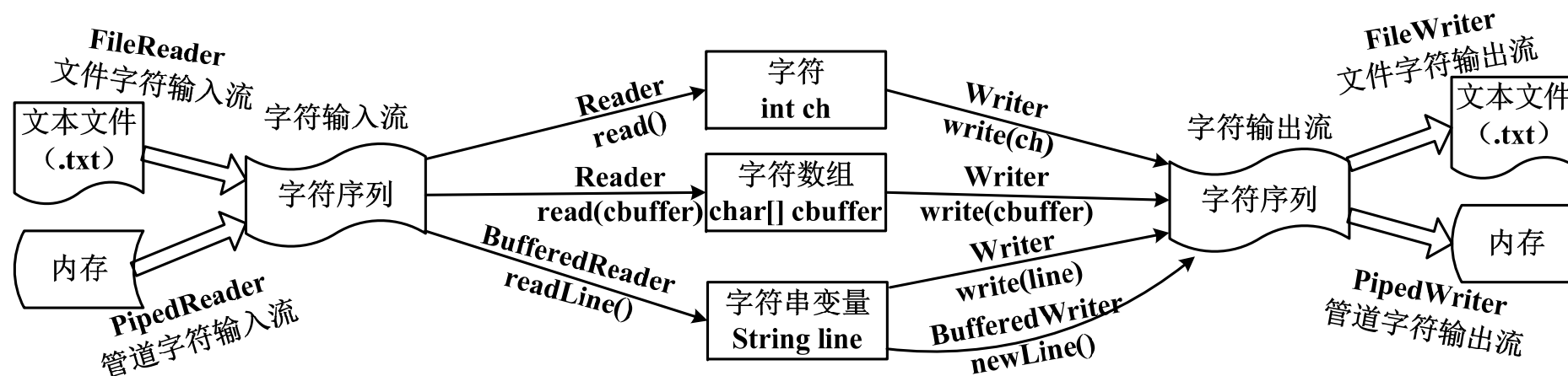


## 2. **Writer**抽象字符输出流类

---

```
public abstract class Writer implements
    Appendable, Closeable, Flushable
{
    public void write(int c) throws IOException
    public void write(char[] cbuffer) throws
        IOException
    public void write(String str) throws IOException
        //将字符串写入输出流
    public abstract void close() throws IOException
}
```

# 图8.10 各种字符输入/输出流的读/写方法





## 8.3.2 字节/字符转换流

1. **InputStreamReader**字节输入流转换成字符输入流类

```
public class InputStreamReader extends Reader  
{
```

```
    public InputStreamReader(InputStream in)
```

```
        //in指定字节输入流作为数据源，采用默认字符集
```

```
    public InputStreamReader(InputStream in,  
        String charsetName)    //指定字符集名称  
        throws UnsupportedEncodingException
```

```
        //不支持charsetName字符集，抛出异常
```

```
    public String getEncoding() //返回字符集名称字符串  
}
```

## 2. 字节输出流转换成字符输出流

### OutputStreamWriter

```
public class OutputStreamWriter extends
    Writer
{
    public OutputStreamWriter(OutputStream
        out) // 指定字节输出流作为数据源，默认字符集
    public OutputStreamWriter(OutputStream
        out, Charset charset) // charset指定字符集
    public String getEncoding() // 返回字符集名
}
```



## 8.3.3 文件字符流

---

### 1. **FileReader**文件字符输入流类

```
public class FileReader extends  
    InputStreamReader  
{  
    public FileReader(String fileName)  
        throws FileNotFoundException  
    public FileReader(File file) throws  
        FileNotFoundException  
}
```





## 2. **FileWriter**文件字符输出流类

---

```
public class FileWriter extends OutputStreamWriter
{
    public FileWriter(String fileName)
                                throws IOException
    public FileWriter(String fileName, boolean append)
                                throws IOException
    public FileWriter(File file) throws IOException
    public FileWriter(File file, boolean append)
                                throws IOException
}
```



## 8.3.4 缓冲字符流

---

### 1. **BufferedReader** 字符缓冲输入流类

```
public class BufferedReader extends Reader
{
    public BufferedReader(Reader reader)
    public String readLine() throws IOException
        // 读取一行字符串，输入流结束时返回null
}
```

### 2. **BufferedWriter** 字符缓冲输出流类

```
public class BufferedWriter extends Writer
{
    public BufferedWriter(Writer writer)
    public void newLine() throws IOException // 写入换行符
}
```



## 8.3.5 格式化字符输出流

```
public class PrintWriter extends Writer
{
    public PrintWriter(OutputStream out, boolean autoFlush)
        //autoFlush指定是否立即传输，默认false
    public void print(boolean bool) //输出boolean取值true或false
    public void print(char ch)
    public void print(int i) //将整数i的取值转换成字符串输出
    public void print(long l)
    public void print(float f)
    public void print(double d)
    public void print(char s[])
    public void print(String s)
    public void print(Object obj) //默认调用obj.toString()方法
    public void println() //自动追加回车换行符，其他重载方法参数同上
}
```



## 【例8.5】使用字符流读写文本文件。

### 1. 例8.2

```
public void readFromText(String filename,  
DefaultTableModel tablemodel)
```

//将从**filename**指定文本文件中读取的字符串，转换成**int**整数，添加到**tablemodel**表格模型中

### 2. 例8.3

//将**listModel**列表框模型中的对象写到**filename**指定文件名的文本文件

```
public <T> void writeToText(String filename,  
DefaultListModel<T> listModel)
```



## 【思考题8-6】

---

### 1. 实现例8.2

//将**tableModel**表格模型中的**int**整数写入由  
**filename**指定文件名的文本文件

```
public void writeToText(String filename,  
DefaultTableModel tablemodel)
```

### 2. 实现例8.3

```
public void readFromText(String filename,  
DefaultListModel<Person> listModel)
```

//将从**filename**指定文件名的对象文件中读取的对  
象添加到**listModel**列表框模型中



## 8.3.6 Java标准输入/输出

---

### 1. 标准输入/输出常量

```
public final class System extends Object
{
    public final static InputStream in //标准输入常量
    public final static PrintStream out //标准输出常量
    public final static PrintStream err
                                   //标准错误输出常量
}
```



# PrintStream 格式化字节输出流 类

---

```
public class PrintStream
    extends FilterOutputStream
{
    public void print(boolean b)
    public void print(char c)
    public void print(long l)
    public void print(int i)
    public void print(float f)
    public void print(double d)
    public void print(String s)
    public void print(Object obj)
    public void println()
}
```



## 【例8.1】标准输入问题讨论。

---

1. **System.in.read()**按字节读取字符，存在问题
2. 使用**String**实现**GBK**编码字符串与**Unicode**编码字符串的双向转换
3. 使用字节/字符转换流





## 8.4 文件操作

---

- 8.4.1 文件类及其过滤器类
- 8.4.2 文件对话框组件
- 8.4.3 随机存取文件类



## 8.4.1 文件类及其过滤器类

### 1. 构造文件和目录对象

```
public class File extends Object implements  
    Serializable, Comparable<File>  
{  
    public File(String pathname)  
    public File(String parent, String child)  
    public File(File parent, String child)  
}
```

```
File file = new File("myfile.txt");  
File dir = new File(".", ""); // 目录文件, 当前目录  
File dir = new File("C:", "");
```



## 2. File类提供的方法

---

### (1) 访问文件对象方法

**String getName()** // 返回文件名，不包含路径名

**String getPath()** // 返回相对路径名，包含文件名

**String getAbsolutePath()**

// 返回绝对路径名，包含文件名

**String getParent()** // 返回父文件对象的路径名

**File getParentFile()** // 返回父文件对象



## (2) 获得或设置文件属性

---

**long length()** // 返回文件的字节长度  
**long lastModified()** // 返回文件的最后修改时间  
**boolean exists()** // 判断当前文件或目录是否存在  
**boolean isFile()** // 判断当前文件对象是否为文件  
**boolean isDirectory()** // 判断文件对象是否为目录  
**boolean setReadOnly()** // 设置文件属性为只读  
**boolean setLastModified(long time)**  
// 设置文件的最后修改时间



## (3) 文件操作方法

---

**int compareTo(File pathname)**

// 比较两个文件对象的内容

**boolean renameTo(File dest)**

// 文件重命名

**boolean createNewFile() throws**

**IOException** // 创建新文件

**boolean delete()** // 删除文件或空目录



## (4) 目录操作方法

---

**public boolean mkdir()**

//创建指定目录，正常建立时返回**true**

**public String[] list()**

//返回目录中的所有文件名字符串

**public File[] listFiles()**

//返回目录中的所有文件对象



## 3. 文件过滤器接口

---

### 1. **FileFilter**接口

```
public interface FileFilter
{
    public boolean accept(File pathname)
}
```

### 2. 获得过滤后的文件列表

```
public File[] listFiles(FileFilter filter)
```



## 【例8.7】 显示带过滤器的文件列表


---

1. 实现文件过滤器接口
2. 音乐播放器的文件列表

**【思考题8-7】** 实现目录文件过滤器，返回当前目录下的所有子目录列表。



## 8.4.2 文件选择对话框组件



```
public class JFileChooser extends JComponent implements Accessible
{
    public static final int APPROVE_OPTION = 0;    //单击“打开”或“保存”按钮
    public static final int CANCEL_OPTION = 1;      //单击“撤消”按钮
    public static final int ERROR_OPTION = -1;      //出错

    public JFileChooser()
    public JFileChooser(String currentDirectoryPath) //初始路径
    public JFileChooser(File currentDirectory)

    public void setFileFilter(FileFilter filter)    //设置文件过滤器
    public int showOpenDialog(Component parent) throws HeadlessException
                                                //显示打开文件对话框
    public int showSaveDialog(Component parent) throws HeadlessException
                                                //显示保存文件对话框
    public File getSelectedFile()                //返回选中文件
}
```



# JFileChooser的文件过滤器

---

```
public abstract class FileFilter
{
    public abstract boolean accept(File f)
        // 过滤操作，f指定待过滤文件
    public abstract String getDescription()
        // 文件类型描述字符串
}
```



## 【例8.8】 文件管理器和文本文件编辑器。

---

1. 文本文件编辑器
2. **JFileChooser**使用的文件过滤器
3. 文件管理器

## 8.4.3 随机存取文件类



```
public class RandomAccessFile extends Object  
    implements DataOutput, DataInput,  
    Closeable
```

```
{
```

```
    public RandomAccessFile(String name,  
        String mode) throws  
        FileNotFoundException
```

```
    public RandomAccessFile(File file, String  
        mode) throws FileNotFoundException
```

//**mode**指定文件访问模式，取值为**r**（只读）、**rw**（读写）。



# RandomAccessFile

**final int readInt() throws IOException**

//读整数，读到文件尾抛出**EOFException**异常

**final void writeInt(int v) throws IOException** //写整

**long length() throws IOException** //文件长度

**long getFilePointer() throws IOException** //文件指针

**void seek(long pos) throws IOException** //文件指针

**void close() throws IOException** //关闭文件

}

**【例8.9】** 使用随机存取文件生成素数文件。

继承例8.2框架



## 实验8 输入/输出流与文件操作

- 目的：使用流和文件操作。
- 要求：
  - ① 掌握字节流和字符流的功能和使用方法；
  - ② 掌握文件操作的基本方法。
- 重点：理解流在文件操作中的作用；熟悉对文件操作的**File**类、文件过滤器、文件对话框和随机存取文件类。
- 难点：①如何选择使用哪种字节流或字符流。  
② 文件组织方式是树结构，递归算法。