Java 程序设计

第12章 流与文件

田英鑫 tyx@hit.edu.cn
哈尔滨工业大学软件学院

第12章 流与文件

& 本章导读

- 12.1 流
- 12.2 完整的流结构
- 12.3 流的使用
- 12.4 文件管理

第12章 流与文件

& 本章重点

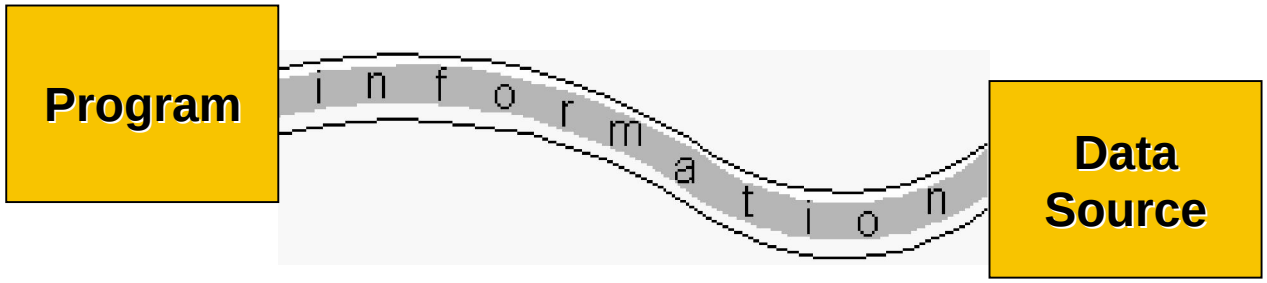
- 12.1 流
- 12.3 流的使用
- 12.4 文件管理

& 本章难点

- 12.1 流
- 12.3 流的使用

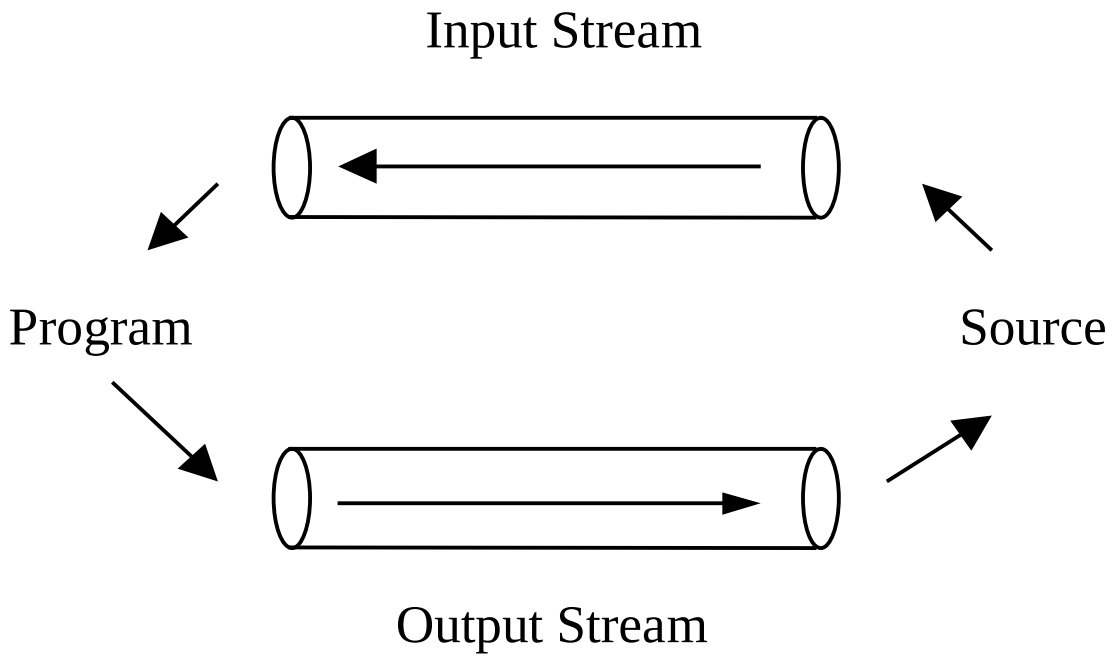
12.1 流

- 流的基本概念
  - 流是一个很形象的概念，当程序需要读取数据的时候，就会开启一个通向数据源的流，这个数据源可以是文件，内存，或是网络连接
  - 类似的，当程序需要写入数据的时候，就会开启一个通向目的地的流。这时候你就可以想象数据好像在这其中“流”动一样



12.1 流

- 流的基本概念
  - 可以读取字节序列的对象称为输入流
  - 可以写入字节序列的对象称为输出流

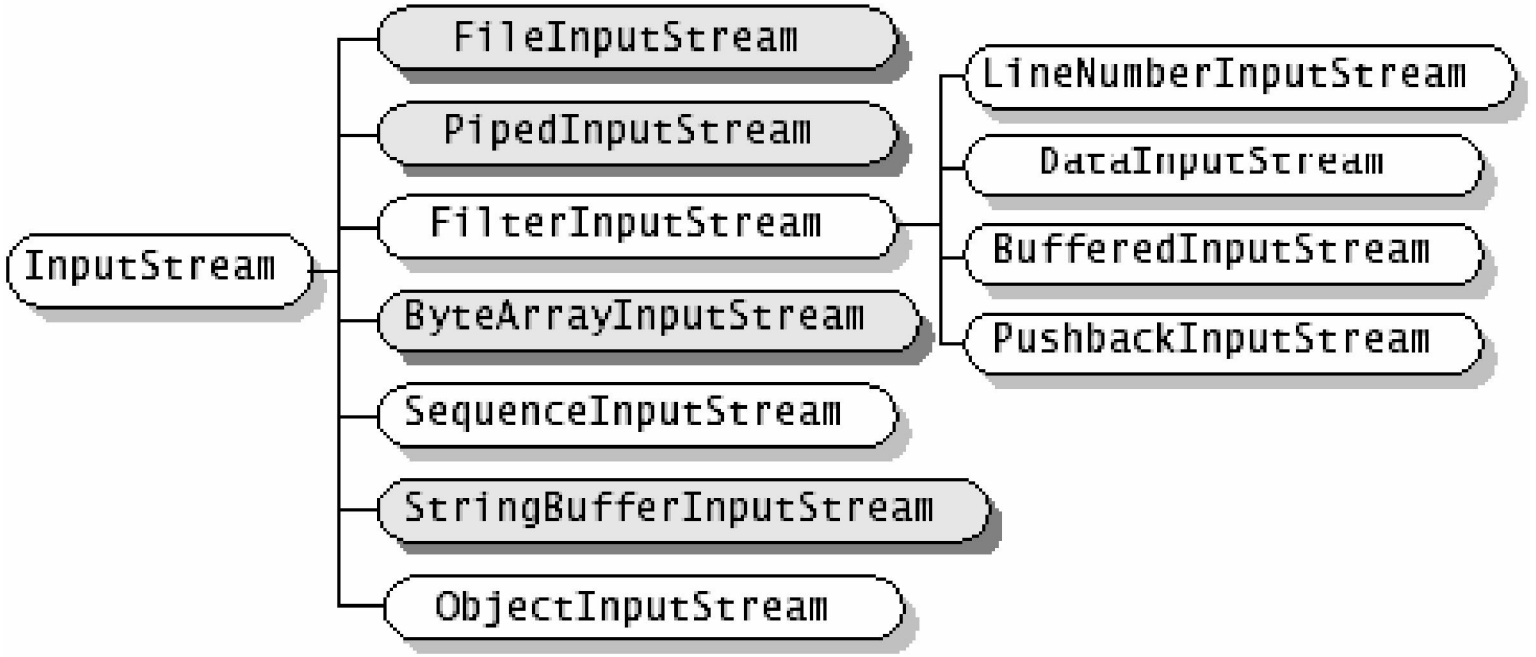


12.1 流

- 流的分类
  - 数据流的类可以分为字节流（byte stream）和字符流（character stream）两种类型，分别由四个抽象基类来表示
  - 字节流的基类
    - 抽象类InputStream
    - 抽象类OutputStream
  - 字符流的基类
    - 抽象类Reader
    - 抽象类Writer

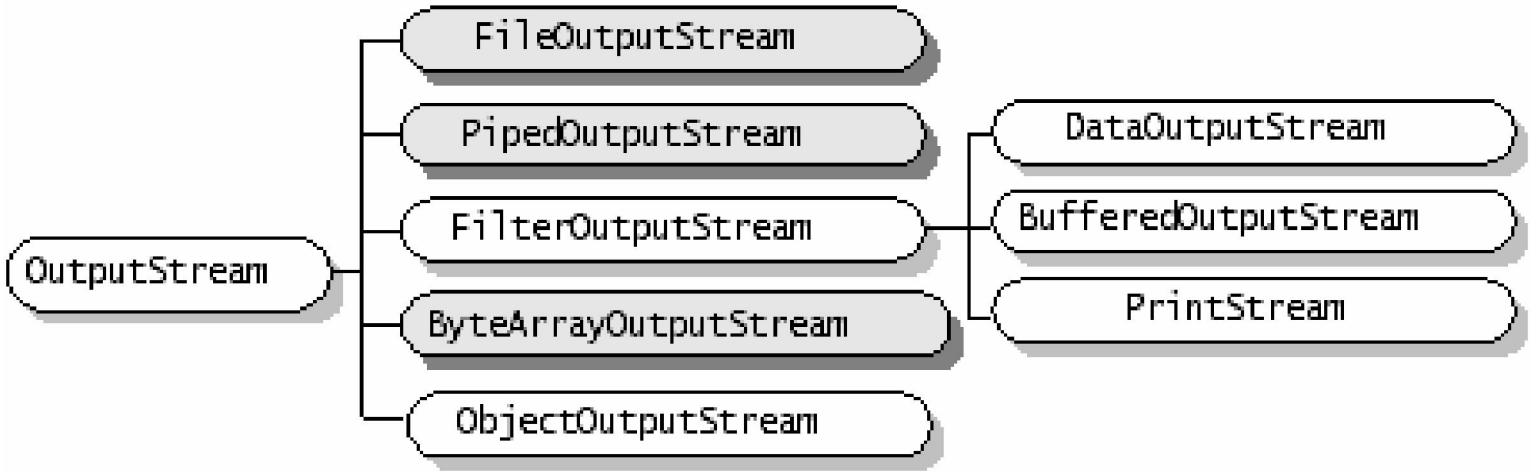
12.1 流

- 流的分类
  - InputStream



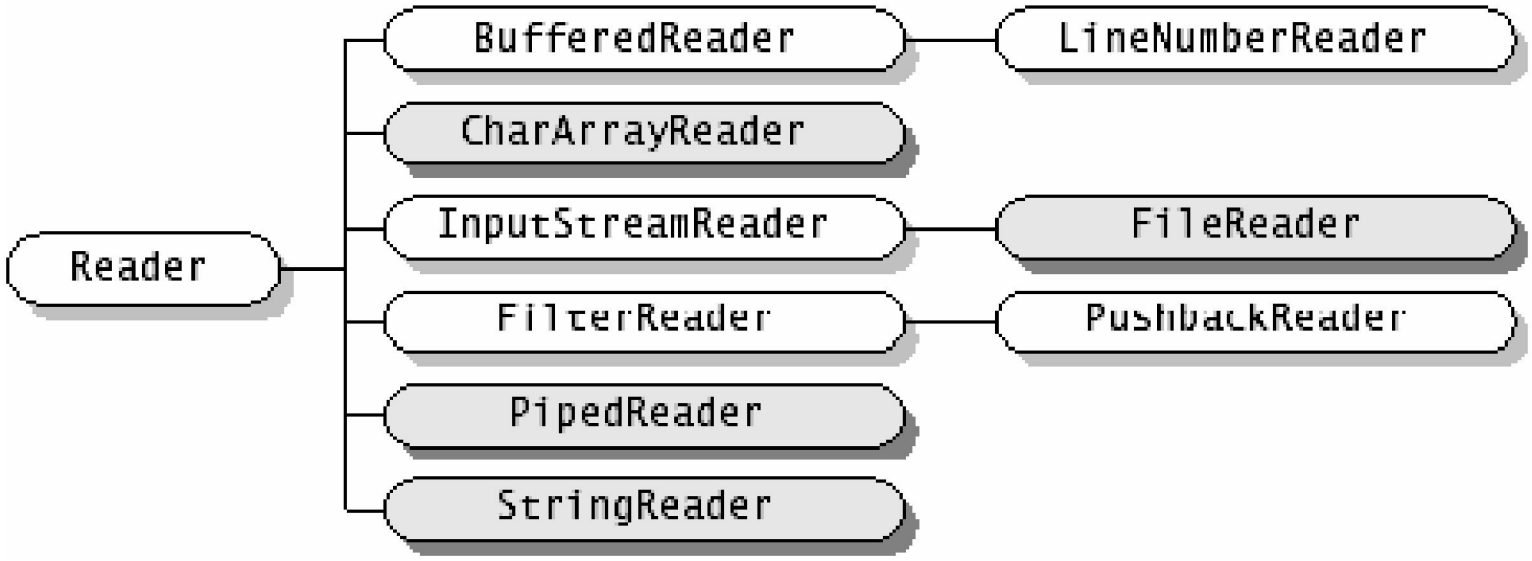
12.1 流

- 流的分类
  - OutputStream



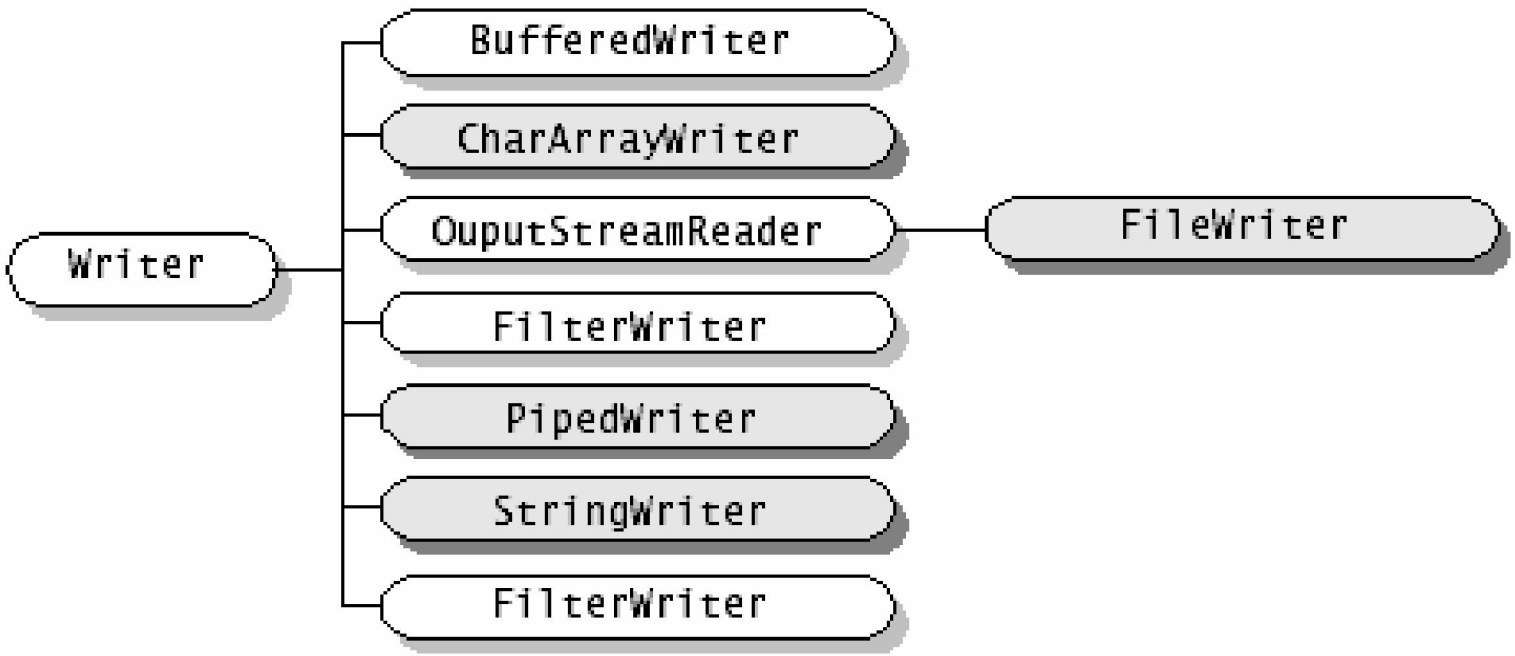
12.1 流

- 流的分类
  - Reader



12.1 流

- 流的分类
  - Writer



12.1 流

- 字节流（InputStream）的读取
  - abstract int read()
    - 读取下一个字节并返回它的整数表示
  - int read(byte b[])
    - 把字节读到数组b中，并返回实际读入的字节数
  - long skip(long n)
    - 跳过输入数据流中n个字节的数据，并返回实际跳过的字节数
  - int available()
    - 该方法返回能够从输入数据流中读取的字节数
  - void close()
    - 该方法关闭数据流

12.1 流

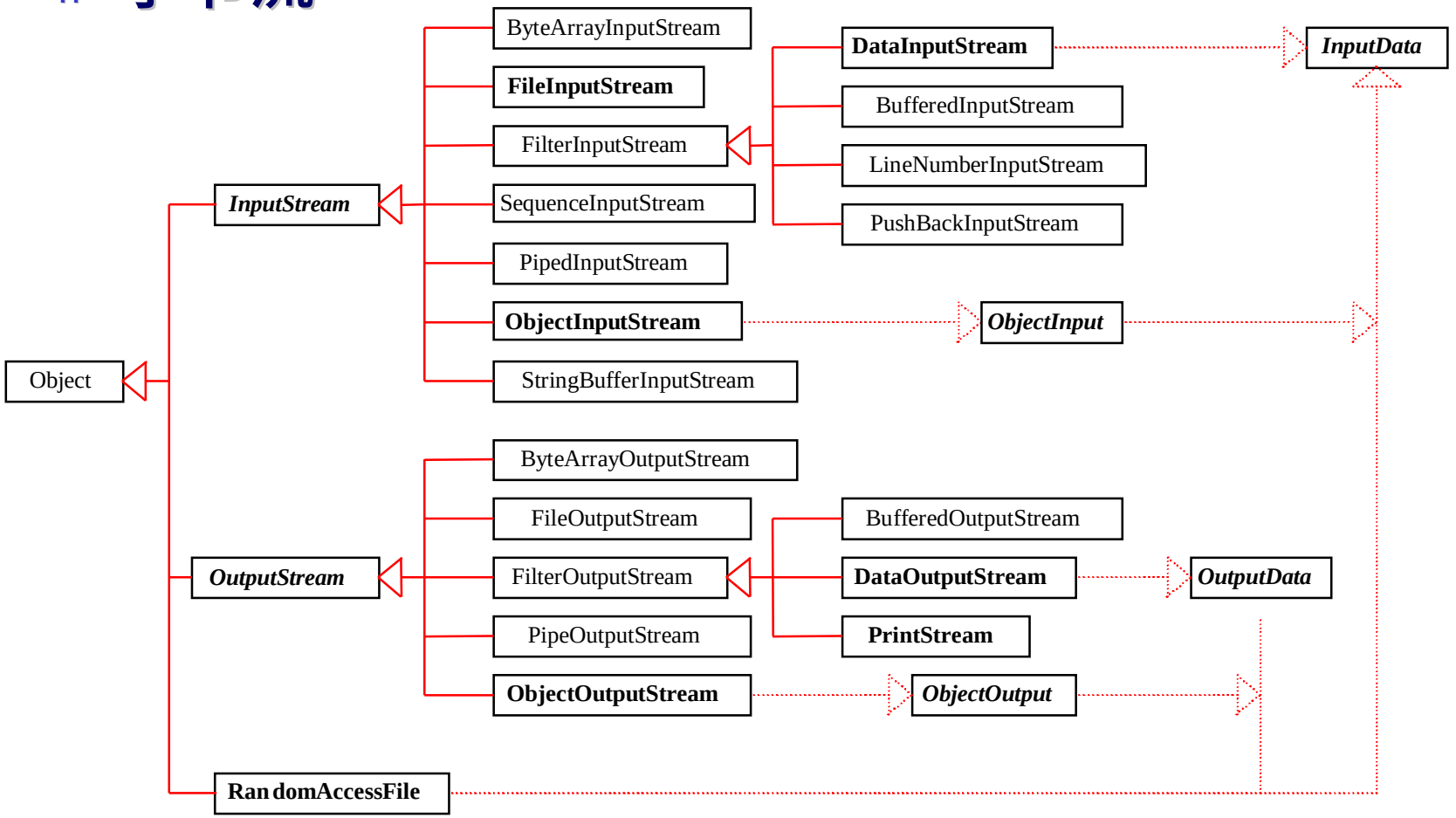
- 字节流（OutputStream）的写入
  - abstract void write(int b)
    - 该方法向输出数据流写入一个字节
  - void write(byte[] b)
    - 该方法把数组b中的全部字节写入到输出数据流
  - void flush()
    - 该方法把数据流中的任何缓冲数据都发送到目的地并清空缓冲区
  - void close()
    - 该方法关闭输出数据流

12.1 流

- 字符流（Reader）的读取
  - abstract int read()
  - int read(char b[])
  - long skip()
  - void close()
- 字符流（Writer）的写入
  - abstract void write(int b)
  - void write(char[] b)
  - void flush()
  - void close()

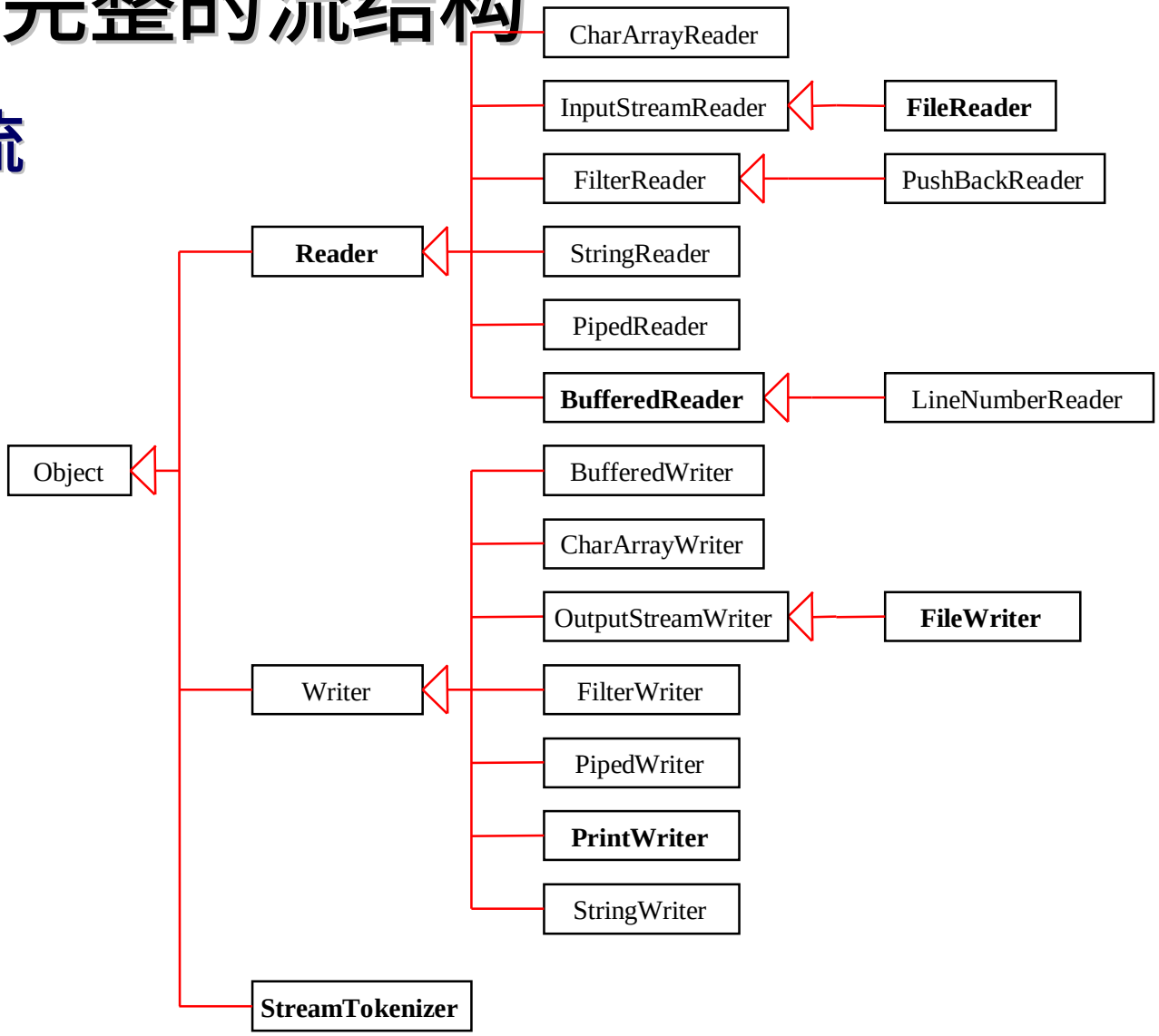
12.2 完整的流结构

- 字节流



12.2 完整的流结构

- 字符流



12.2 完整的流结构

- 流过滤器的分层
  - 对某些流来说，可以从文件或者其他地方读入字节，而对另一些流来说，它们则可将字节“组装”成更有用的数据类型
  - Java程序员通过将一个现成的流传递给另一个流的构造器来综合运用这两种流，将其合并成所谓的“过滤流”
    - 例如：为了能从文件中读取数字，首先要创建一个FileInputStream对象，然后将其传递给一个DataInputStream的构造器

```
FileInputStream fin = new FileInputStream("employee.dat");
DataInputStream din = new DataInputStream(fin);
double s = din.readDouble();
```

12.2 完整的流结构

- 数据流
  - 我们经常需要将计算结果写入流，或者从流中读回结果
  - 数据流支持读取和写入基本Java类型的方法
  - DataInputStream
    - 该类实现了DataInput接口中对各种基本类型的读取方法
  - DataOutputStream
    - 该类实现了DataOutput接口中对各种基本类型的写入方法

12.2 完整的流结构

- 数据流
  - DataInputStream
    - int readByte()
    - int readShort()
    - int readInt()
    - int readLong()
    - float readFloat()
    - double readDouble()
    - char readChar()
    - boolean readBoolean()

12.2 完整的流结构

- 数据流
  - DataOutputStream
    - void writeByte(byte b)
    - void writeShort(short s)
    - void writeInt(int i)
    - void writeLong(long l)
    - void writeFloat(float f)
    - void writeDouble(double d)
    - void writeChar(char c)
    - void writeBoolean(boolean b)

12.2 完整的流结构

- 数据流
  - 构造输入/输出数据流

```
// 为文件in.dat创建一个输入流
DataInputStream infile = new DataInputStream
(
   new FileInputStream("in.dat")
);

// 为文件out.dat 创建一个输出流
DataOutputStream outfile = new DataOutputStream
(
   new FileOutputStream("out.dat")
);
```

参见:TestDataStreams.java

12.2 完整的流结构

- 随机存取文件流
  - 随机存取文件流（RandomAccessFile）可以在文件的任何地方查找或写入数据
  - RandomAccessFile类的构造方法

```
// 以只读方式打开文件
RandomAccessFile raf =
new RandomAccessFile("test.dat","r");

// 以读写方式打开文件
RandomAccessFile raf =
new RandomAccessFile("test.dat","rw");
```

12.2 完整的流结构

- 随机存取文件流
  - RandomAccessFile类的方法
    - boolean readBoolean()
    - char readChar()
    - byte readByte()
    - short readShort()
    - int readInt()
    - long readLong()
    - float readFloat()
    - double readDouble()
    - String readLine()

12.2 完整的流结构

- 随机存取文件流
  - RandomAccessFile类的方法
    - void writeBoolean(boolean b)
    - void writeChar(int v)
    - void writeByte(int v)
    - void writeShort(int v)
    - void writeInt(int v)
    - void writeLong(long v)
    - void writeFloat(float v)
    - void writeDouble(double v)
    - void writeChars(String s)
    - void write(byte[] b)

12.2 完整的流结构

- 随机存取文件流
  - RandomAccessFile类的方法
    - void seek(long pos)
      - 将偏移设置为从数据流的开始处到下一次读写的位置
    - long getFilePointer()
      - 返回偏移的的字节值，偏移是指从文件开始到下一次读写位置的距离
    - long length()
      - 返回文件的长度

参见:TestRandomAccessFile.java

12.2 完整的流结构

- 文本输出
  - PrintWriter
  - 构造方法
    - PrintWriter(Writer out)
    - PrintWriter(Writer out, boolean autoFlush)
    - PrintWriter(OutputStream out)
    - PrintWriter(OutputStream out, boolean autoFlush)
  - 常用方法
    - void print(…)
    - void println(…)

12.2 完整的流结构

- 文本输出
  - 输出到文件

```
import java.io.*;
class Test
{
  public static void main(String[] args) throws IOException
  {
    FileOutputStream fos = new FileOutputStream("employee.txt");
    PrintWriter pw = new PrintWriter(fos);
    pw.println(100);
    pw.println("abc");
    pw.println(false);
    pw.flush();
    pw.close();
    fos.close();
  }
}
```

12.2 完整的流结构

- 文本输入
  - BufferedReader

```
// 从文件输入
BufferedReader in = new BufferedReader(
    new FileReader("employee.txt")
);
// 获取键盘输入
BufferedReader br = new BufferedReader(
    new InputStreamReader(System.in)
);

// 常用方法
in.readLine();
```

12.2 完整的流结构

- 文本输入
  - 获取键盘输入

```
import java.io.*;
public class MyInput {
    public static String readString() {
        BufferedReader br = new BufferedReader(
           new InputStreamReader(System.in), 1);
        String string = " ";
        try {
           string = br.readLine();
        }
        catch (IOException ex) {
           System.out.println(ex);
        }
        return string;
   }
}
```

参见:MyInput.java

12.3 流的使用

n 分隔符输出

```
// 按如下方式存储对象的状态
Harry Hacker|35500|1989|10|1
Carl Cracker|75000|1987|12|15
Tony Tester|38000|1990|3|15
// 向Employee类添加如下writeData方法
public void writeDate(PrintWriter out) throws IOException
{
   GregorianCalendar calendar = new GregorianCalendar();
   calendar.setTime(hireDay);
   out.println(name + "|" + salary + "|"
      + calendar.get(Calendar.YEAR) + "|"
      + (calendar.get(Calendar.MONTH) + 1) + "|"
      + calendar.get(Calendar.DAY_OF_MONTH));
}
```

12.3 流的使用

- 字符串记号处理器和带分隔符的文本
  - StringTokenizer类

```
// 构造StringTokenizer对象,可以指定一个或多个分隔符
StringTokenizer tokenizer = new StringTokenizer(line,"|");
StringTokenizer tokenizer = new StringTokenizer(line,"|,;");

// 常用方法
public boolean hasMoreTokens()
// 如果字符串中还有令牌则该方法返回true

public String nextToken()
// 该方法返回串中下一个令牌

public int countTokens()
// 返回令牌的总数
```

参见:TestStringTokenizer.java

12.3 流的使用

n 读取带分隔符的输入

```
// 向Employee类添加如下readData方法
public void readDate(BufferedReader in) throws IOException
{
   String s = in.readLine();
   StringTokenizer t = new StringTokenizer(s,"|");
   name = t.nextToken();
   salary = Double.parseDouble(t.nextToken());
   int y = Integer.parseInt(t.nextToken());
   int m = Integer.parseInt(t.nextToken());
   int d = Integer.parseInt(t.nextToken());
   GregoriznCalendar calendar = new GregorianCalendar(y,m-1,d);
   hireDay = calendar.getTime();
}
```

参见例12-2:DataFileTest.java

12.3 流的使用

- StringBuilder 类
  - 动态字符串

12.3 流的使用

- 随机存取流

参见例12-3:RandomFileTest.java

12.4 文件管理

- File 类
  - File类是文件名的一个包装类
    - File file = new File("test.txt");
  - File类的常用方法
    - exists() 查看文件是否存在
    - getName() 得到文件名
    - getPath() 得到文件完整路径
    - getParent() 得到包含文件的目录
  - 必须使用文件数据流对磁盘文件进行读写
    - FileInputStream和FileOutputStream用于字节流
    - FileReader和FileWriter用于字符流

12.4 文件管理

File 类

```
File f = new File("c:\\student.txt");
if(f.exists()) {
  f.delete();
}
try {
  f.createNewFile();
}
catch(Exception e) {
  System.out.println(e.getMessage());
}
System.out.println(f.getName());
System.out.println(f.getPath());
System.out.println(f.getAbsolutePath());
System.out.println(f.getParent());
System.out.println(f.canRead());
System.out.println(f.lastModified());
System.out.println(f.length());
```

参见:TestFile.java

Any Question?