



第十三章 流与文件（一）

讲授思路

- 文件处理
- I/O和流
- 字节流

讲授思路-文件处理

- 文件的基本概念
- Java中的文件操作
 - 得到信息
 - 创建
 - 删除
 - 修改

文件的基本概念

- Java程序可访问的最主要的外部资源之一就是文件。
- 在Java中用File类来进行文件及目录的操作，常见操作：
 - 查询文件信息
 - 创建文件
 - 文件内容读取
 - 写文件
 - ...

Java中的File类

- Java中的File在Java.io包中，常用的方法有：
 - 参考JDK_API 1.8.0 文档
- 文件操作
 - `boolean canWrite();`
 - `boolean canRead();`
 - `boolean isFile();`
 - `boolean isDirectory();`
 - `long lastModified();`
 - `long length();`
 - `boolean delete();`
- 目录操作
 - `boolean mkdir ();`
 - `String list();`

文件操作—文件信息获得

- 目录是否存在
- 文件是否存在

```
//定义目录对象
File dir = new File("C://Auto//");
boolean r = false;
//判断此目录是否存在
if (!dir.exists()) {
    //如果不存在就创建
    r = dir.mkdir();
    if (r) {
        System.out.println("目录创建成功！");
    }
}
```

```
//创建文件
File f = new File("C://Auto//a.txt");
if (!f.exists()) {
    r = false;
    try {
        r = f.createNewFile();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    if (r) {
        System.out.println("文件创建成功！");
    }
}
```

文件操作—文件创建

- 文件创建
- 目录创建

```
//定义目录对象
File dir = new File("C://Auto//");
boolean r = false;
//判断此目录是否存在
if (!dir.exists()) {
    //如果不存在就创建
    r = dir.mkdir();
    if (r) {
        System.out.println("目录创建成功！");
    }
}
```

```
//创建文件
File f = new File("C://Auto//a.txt");
if (!f.exists()) {
    r = false;
    try {
        r = f.createNewFile();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    if (r) {
        System.out.println("文件创建成功！");
    }
}
```

文件操作—文件的读写

- 文件读写
- 目录中文件列表

```
//得到目录的文件列表
String[] fs = dir.list();
for (int i = 0; i < fs.length; i++) {
    System.out.println(fs[i]);
}
```

```
//写入文件
try {
    Writer w = new FileWriter(f);
    w.write("hello world");
    w.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```


文件操作—文件删除

- 文件删除
- 目录删除

```
//删除文件  
r = f.delete();  
if (r) {  
    System.out.println("文件删除成功!");  
}
```

文件操作—随机存取文件

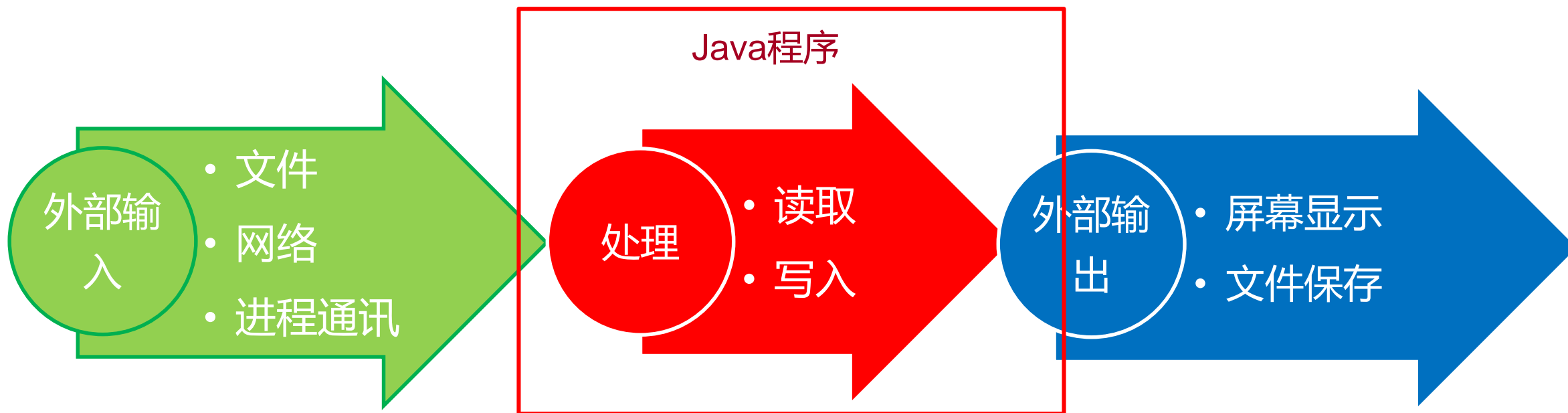
- RandomAccessFile类封装了字节流操作，方便实现随机读取文件的某一个部分。
 - 此类只能操纵文件，通过文件的路径或File对象构造
 - 可以对文件进行读写操作，实例化时指明读写模式
- 常用的方法有：
 - long getFilePointer();
 - void seek(long pos);
 - long length();
 - int read()
 - boolean readBoolean()
 - char readChar()
 - void writeBoolean()
 - void writeByte(int v)
 -

讲授思路-I/O和流

- 流的概念
- 流的作用
- 流的分类
- 流的处理相关类

流的作用

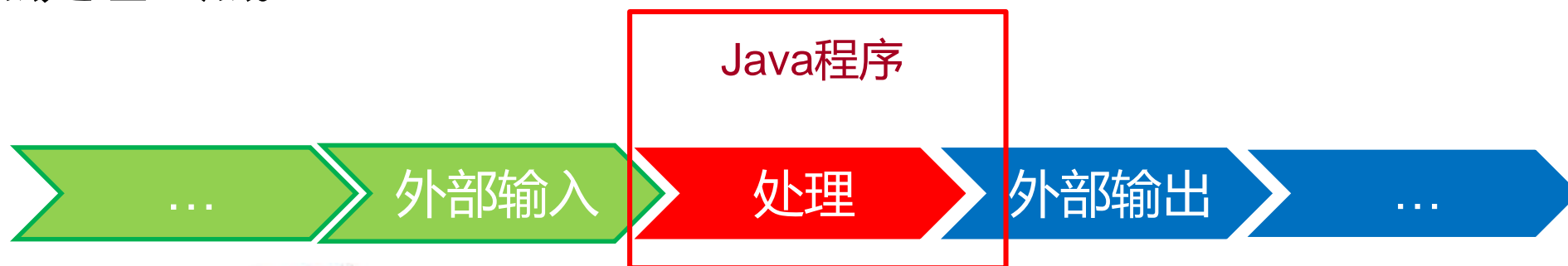
- 流：代表任何有能力产出数据的数据源对象或者有能力接收数据的接收端对象
- 程序如何与外部资源进行交互呢？
- 例如：我希望读取本地的一个文本文件显示在屏幕上怎么办呢？



Java程序与外部交互的主要手段就是流！！

流的基本概念

- 在Java中使用为什么输入输出要用“流”这种方式呢？
- 因为程序输入输出的基本单位是字节，输入就是获取一串字节，输出就是发送一串字节。但是很多情况下，程序不可能接收所有的字节之后再进行处理，而是接收一点处理一点。



YOUKU 优酷
世界都在看

土豆网
tudou.com

流的分类

根据流的方向分类

- 输入流：可以从流中读取信息
- 输出流：可以向流中写入信息

根据操纵对象的类型分类

- 字符流：读取单位为**字符**，因为数据编码的不同，而有了对字符进行高效操作的流对象。本质其实就是基于字节流读取时，去查了指定的码表。读取的字节数由字符编码确定。
- 字节流：读取单位为**字节**，一般用于文件传输

只要处理纯文本数据，优先考虑字符流，其他的形式采用字节流。

Java中流的处理

- 在Java中对于不同的流提供了相应的处理类：
 - 字节输入流：InputStream类
 - 字节输出流：OutputStream类
 - 字符输入流：Reader类
 - 字符输出流：Writer类

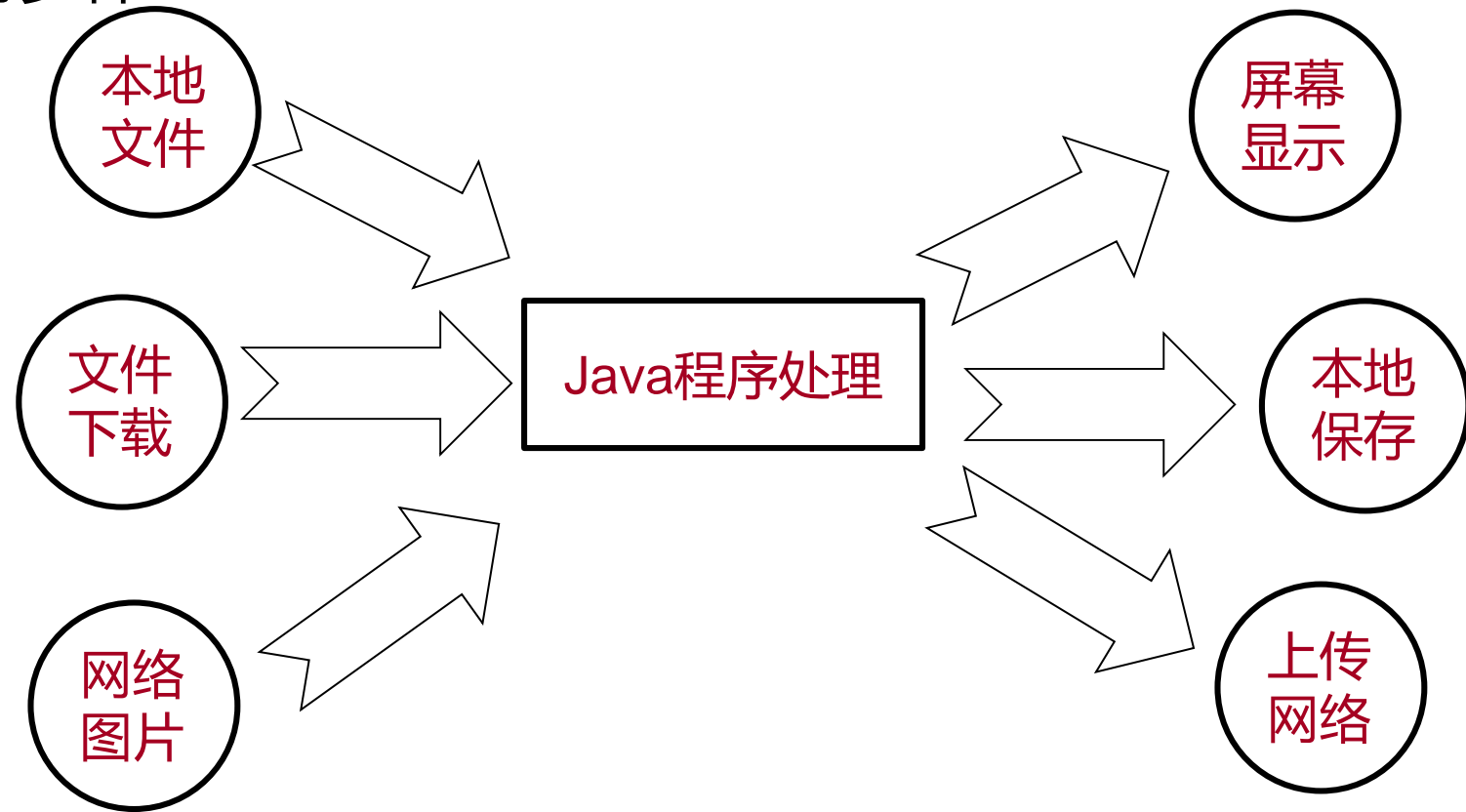
	字节流	字符流
输入流	InputStream	Reader
输出流	OutputStream	Writer

讲授思路-字节流

- 字节流简介
- 输入字节流InputStream
- 输出字节流OutputStream

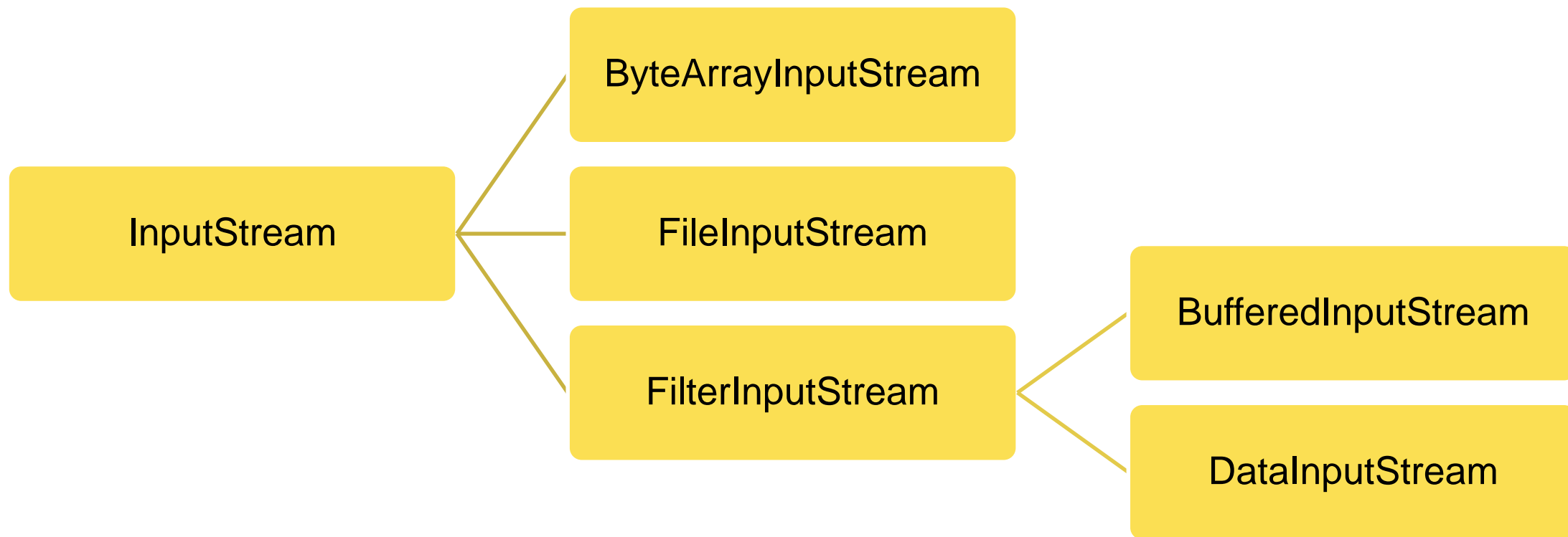
Java中字节流的处理

- 字节流的输入输出主要以字节为单位。主要用于外部设备中文件的读取和外部设备中文件的显示。
- 外部文件的类型有多种：



Java中字节输入流处理InputStream

- Java中所有的字节输入流都用InputStream表示，读取单位为1字节（8位）。
- 为了方便不同外部资源输入，Java中对于不同的外部资源对应了不同的输入流类，他们都是继承自InputStream。



Java中字节输入流处理InputStream

- Java中InputStream的常用方法：
 - `int read();`从输入流中读取数据的下一个字节。
 - `int read(byte c[]);`从输入流中读取一定数量的字节，并将其存储在缓冲区数组 `b` 中。
 - `int read(byte c[], int off, int len);`将输入流中最多 `len` 个数据字节读入 `byte` 数组。
 - `void mark(int readlimit);`在此输入流中标记当前的位置。
 - `void reset();`将此流重新定位到最后一次对此输入流调用 `mark` 方法时的位置。
 - `void close();`关闭此输入流并释放与该流关联的所有系统资源。
- 参考JDK_API 文档

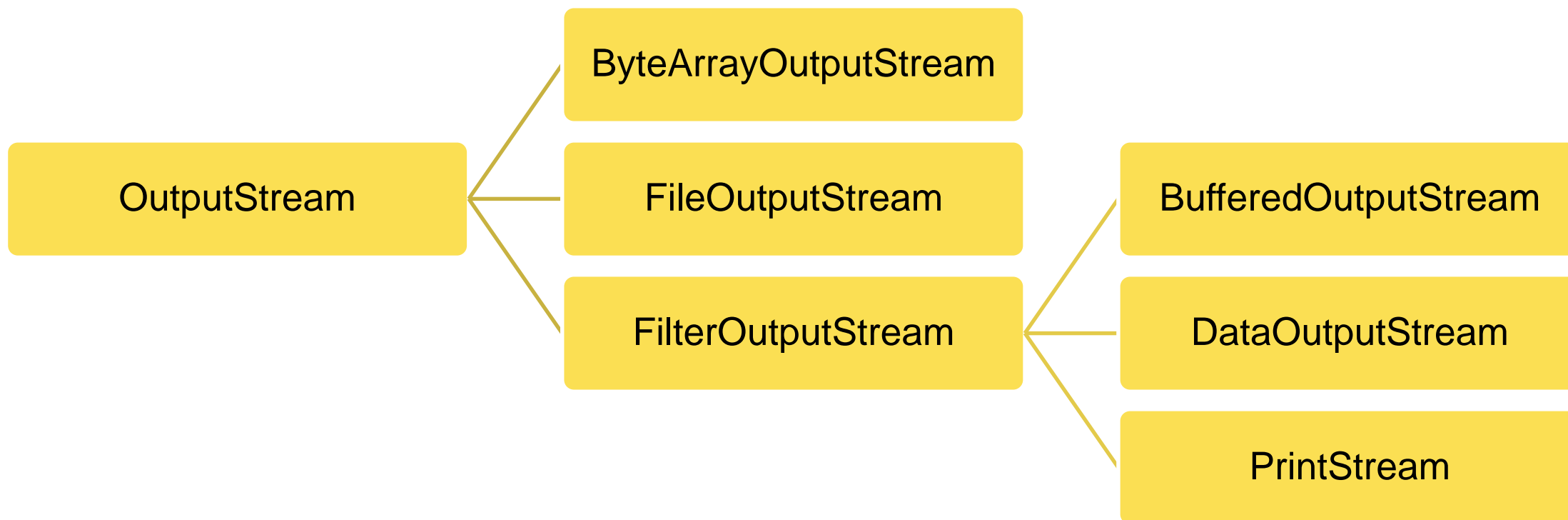
课堂练习：

- 读取本地路径的一个文件C:/a.jpg

```
InputStream input = new FileInputStream(new File("C://testpic.jpg"));  
int i = 0;  
while (i != -1) {  
    i = input.read();  
}  
input.close();
```

Java中字节输出流处理OutputStream

- Java中所有的字节输出流都用OutputStream表示，写入单位为1字节（8位）
- 为了方便输出不同外部资源，Java中对于不同的外部资源对应了不同的输出流类，他们都是继承自OutputStream



Java中字节输出流处理OutputStream

- Java中OutputStream的常用方法：
 - `void write(int b);`将指定的字节写入此输出流。
 - `void write(byte b[]);`将 `b.length` 个字节从指定的 `byte` 数组写入此输出流。
 - `void write(byte b[], int off, int len);`将指定 `byte` 数组中从偏移量 `off` 开始的 `len` 个字节写入此输出流。
 - `void flush();`刷新此输出流并强制写出所有缓冲的输出字节。
 - `boolean markSupported();`测试此输入流是否支持`mark` 和 `reset` 方法
 - `void close();`关闭此输出流并释放与此流有关的所有系统资源。
- 参考JDK_API 文档

课堂练习：

- 读取本地路径的一个文件C:/a.jpg
- 写入本地路径的一个文件D:/c.jpg

```
InputStream input = new FileInputStream(new File("C://testpic.jpg"));
OutputStream output = new FileOutputStream(new File("C://resultpic.jpg"));
int i = 0;
while (i != -1) {
    i = input.read();
    output.write(i);
}
input.close();
output.close();
```


总结

- 文件处理
- I/O和流
- 字节流



Thank You