# 9_10流

# CONTENTS

# 本章技能点列表

| 技能点名称 | 难易程度 | 认知程度 | 重要程度 |
| --- | --- | --- | --- |
| IO流的原理 | 易 | 理解 | ** |
| IO流的分类 | 易 | 记忆 | ** |
| 文件流 | 中 | 应用 | *** |
| 缓冲流 | 中 | 应用 | *** |
| 转换流 | 中 | 应用 | ** |
| 打印流 | 中 | 应用 | ** |
| 数组流 | 中 | 应用 | ** |
| 数据类 | 中 | 应用 | *** |
| 序列化和反序列化 | 中 | 理解 | ** |
| 对象流 | 中 | 应用 | *** |
| Properties读写属性文件 | 易 | 应用 | *** |
| 编码和解码 | 中 | 理解 | ** |
| 使用IO流复制文件夹 | 难 | 应用 | *** |
| 使用IO完成商品的添加和查询 | 中 | 应用 | *** |

# File类

- 文件和目录路径名的抽象表示形式。一个File对象可以代表一个文件或目录

- 可以实现获取文件和目录属性等功能
- 可以实现对文件和目录的创建、删除等功能

- File不能访问文件内容

```
File  file = new File("d:\\test\\java.txt");
File  file = new File("d:/test/java.txt");
File  file = new File("java.txt");
```

路径可以是绝对路径和相对路径，分隔符采用\\或者/

# File类

- 通过File对象可以访问文件的属性。

  public boolean canRead()　public boolean canWrite()

  public boolean exists()　public boolean isDirectory()

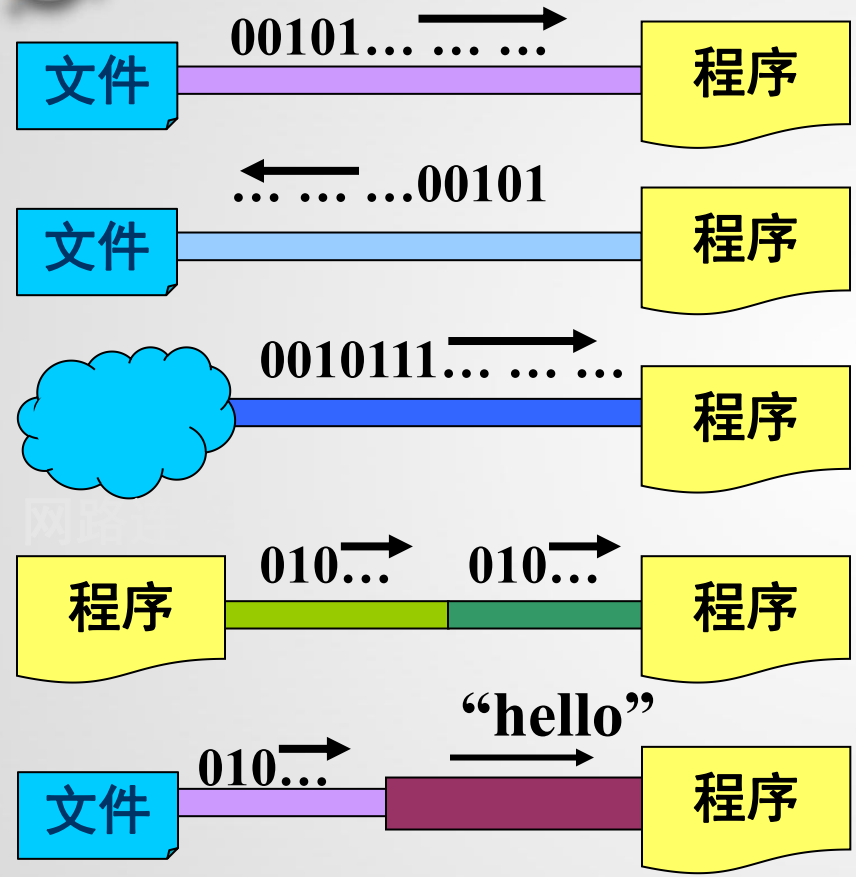
  public boolean isFile()　public boolean isHidden()

  public long lastModified()　public long length()

  public String getName()　public String getPath()

- 通过File对象创建空文件或目录（在该对象所指的文件或目录不存在的情况下）。
  - public boolean createNewFile()throws IOException
  - public boolean delete()
  - public boolean mkdir()， mkdirs()　注意两个的区别！！

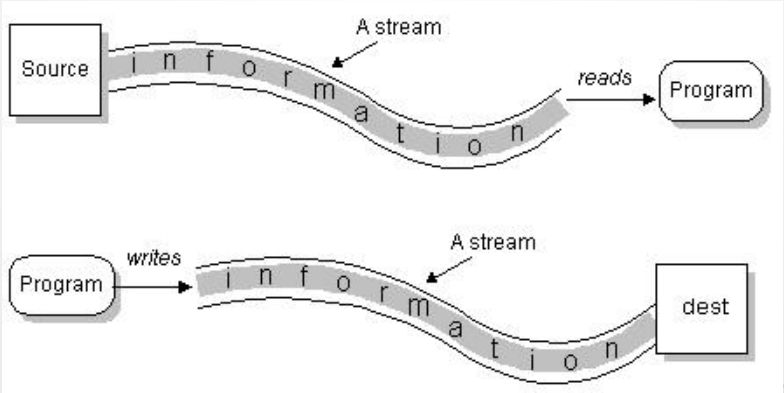
# 流的原理




- 在Java程序中，对于数据的输入/输出操作以“流” (stream) 方式进行；
- J2SDK提供了各种各样的“流”类，用以获取不同种类的数据；程序中通过标准的方法输入或输出数据。
- Java的流类型一般位于java.io包中

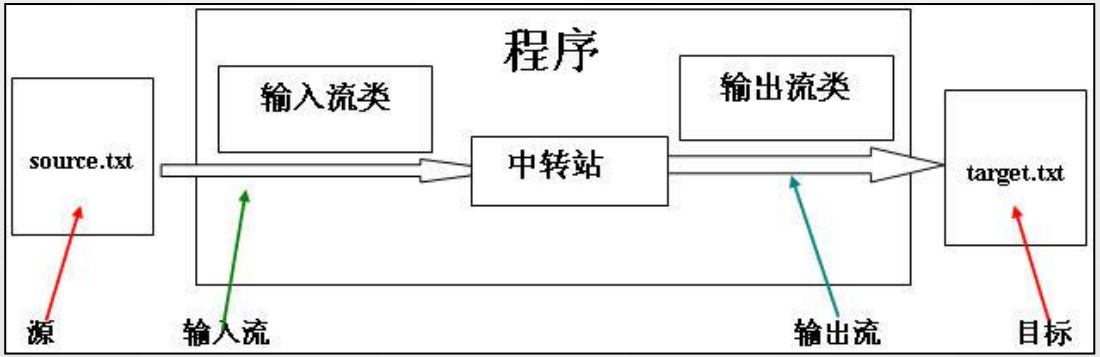
# 流的概念

- 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。



# 流的分类

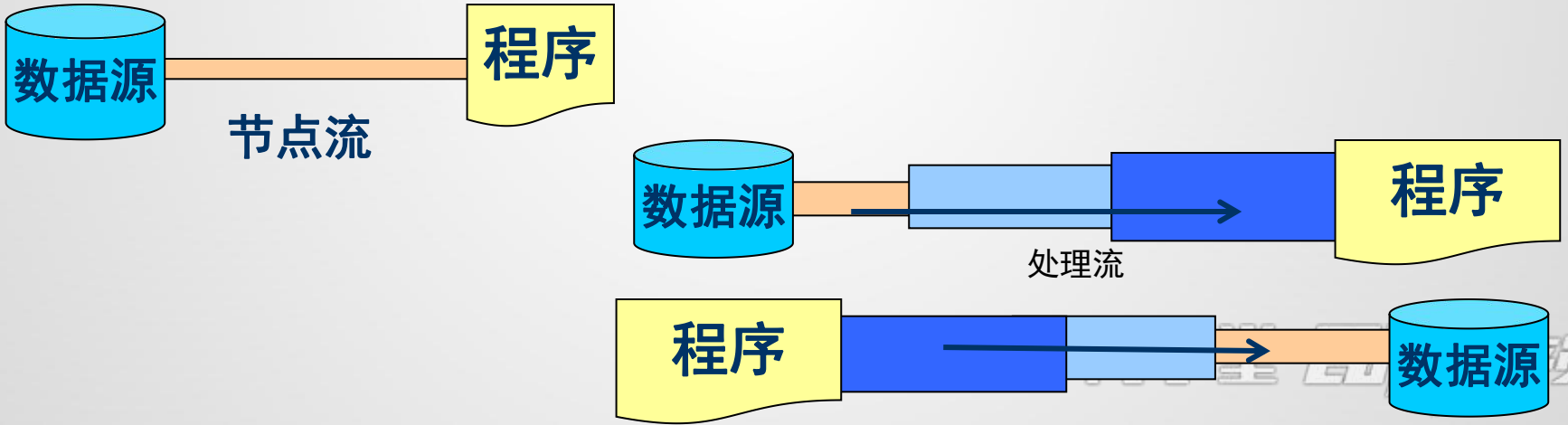
- 流的方向：
  - 输入流：数据源到程序(InputStream、Reader读进来)
  - 输出流：程序到目的地(OutPutStream、Writer写出去)
- 处理数据单元：
  - 字节流：按照字节读取数据(InputStream、OutputStream)
  - 字符流：按照字符读取数据(Reader、Writer)




输入输出是相对于程序而言，而不是相对于源和目标而言

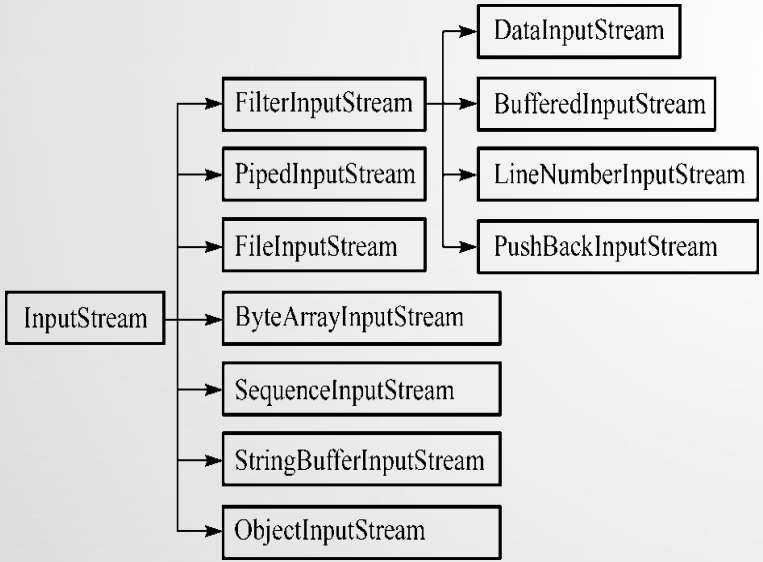
# 流的分类

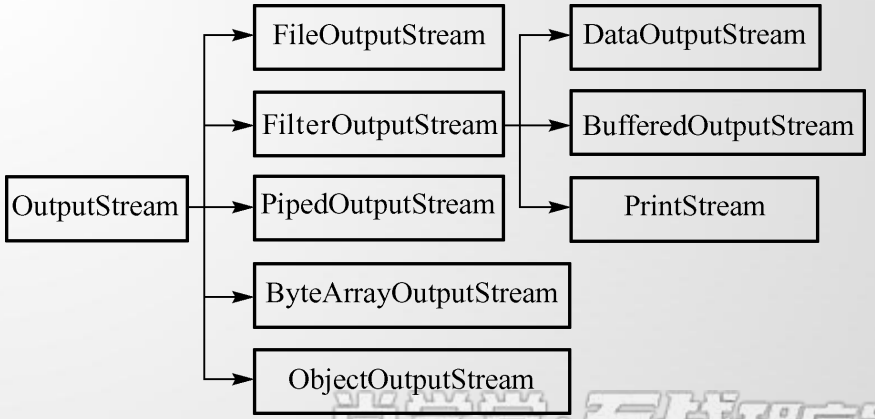
- 功能不同：
  - 节点流：可以直接从数据源或目的地读写数据。
  - 处理流(包装流)：不直接连接到数据源或目的地，是其他流进行封装。目的主要是简化操作和提高性能.
- 节点流和处理流的关系：
  - 节点流处于io操作的第一线，所有操作必须通过他们进行；
  - 处理流可以对其他流进行处理(提高效率或操作灵活性).



# IO流类的体系

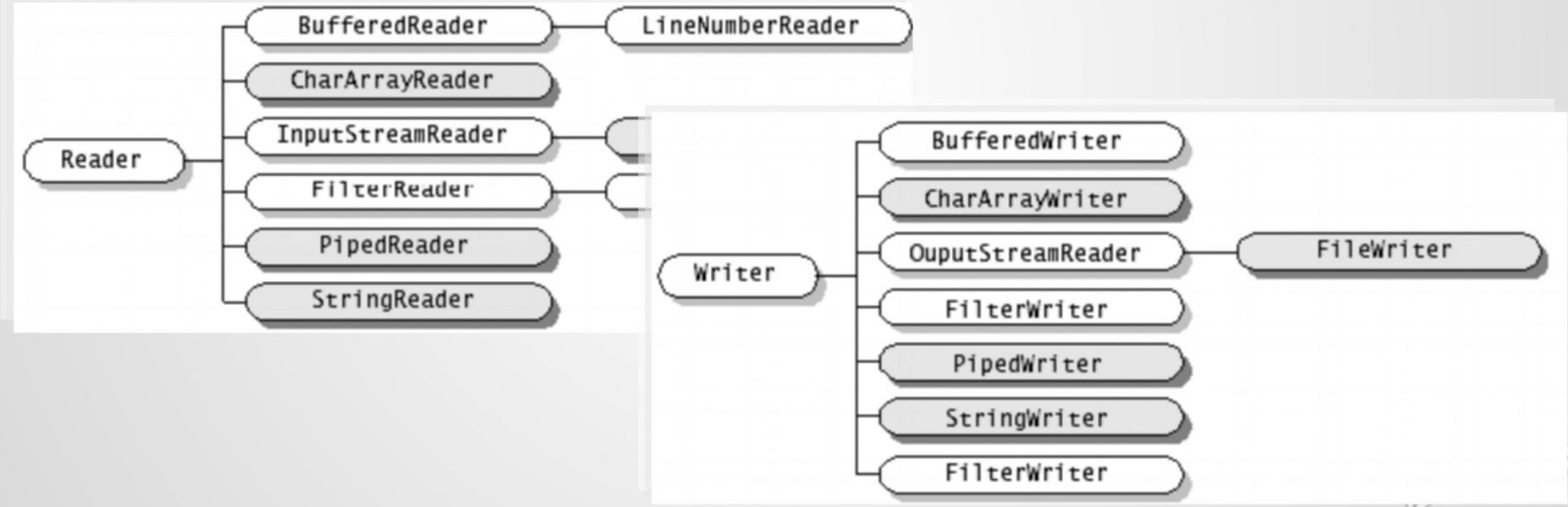
- InputStream和OutputStream
  - Java语言中最基本的两个字节输入输出类。
  - 其他所有字节输入输出流类都继承自这两个基类。
  - 这两个类都是抽象类，不能创建它们的实例，只能使用它们的子类.



# IO流类的体系

- Reader和Writer
  - Java语言中最基本的两个字符输入输出类。
  - 其他所有字符输入输出流类都继承自这两个基类。
  - 这两个类都是抽象类，不能创建它们的实例，只能使用它们的子类.



# FileInputStream/FileOutputStream

- 使用FileInputStream读取文件内容
  - abstract int read( );
  - int read( byte b[ ] );
  - int read( byte b[ ], int off, int len );
  - int available( );
  - close( );
- 使用FileOutputStream写内容到文件
  - abstract void write( int b );
  - void write( byte b[ ] );
  - void write( byte b[ ], int off, int len );
  - void flush( );
  - void close( );
- 使用FileInputStream/FileOutputStream复制文件

# FileReader和FileWriter

- 使用FileReader和FileWriter完成文件复制

# 缓冲字节流

- BufferedInputStream和BufferedOutputStream
  - FileInputStream和FileOutputStream是节电流
  - BufferedInputStream和BufferedOutputStream是处理流（包装流）
  - 读文件和写文件都使用了缓冲区，减少了读写次数，从而提高了效率

  - 当创建这两个缓冲流的对象时时，会创建了内部缓冲数组，缺省使用32字节大小的缓冲区.
  - 当读取数据时，数据按块读入缓冲区，其后的读操作则直接访问缓冲区
  - 当写入数据时，首先写入缓冲区，当缓冲区满时，其中的数据写入所连接的输出流。使用方法flush()可以强制将缓冲区的内容全部写入输出流

  - 关闭流的顺序和打开流的顺序相反.只要关闭高层流即可，关闭高层流其实关闭的底层节点流
  - Flush的使用：手动将buffer中内容写入文件

# 缓冲字符流

- BufferedReader
  - readLine() 读取一个文本行的数据
- BufferedWriter
  - newLine(); 写入一个行分隔符。
- 使用缓冲字符流是复制文本文件常用的方式

```
String str = br.readLine();
while(str != null){
    bw.write(str);
    bw.newLine();
    str = br.readLine();
}
```

# 转换流

- InputStreamReader和OutputStreamWriter
  - 为处理流
  - 用于将字节流转化成字符流，字符流与字节流之间的桥梁
  - InputStreamReader的作用是把InputStream转换成Reader
  - OutputStreamWriter的作用是把OutputStream转换成Writer

  - 存在将字节流转换成字符流的转换流，因为字符流操作文本更简单
  - 不存在把字符流转换成字节流的转换流，因为没有必要
  - System.in代表标准输入，即键盘输入，是InputStream的实例

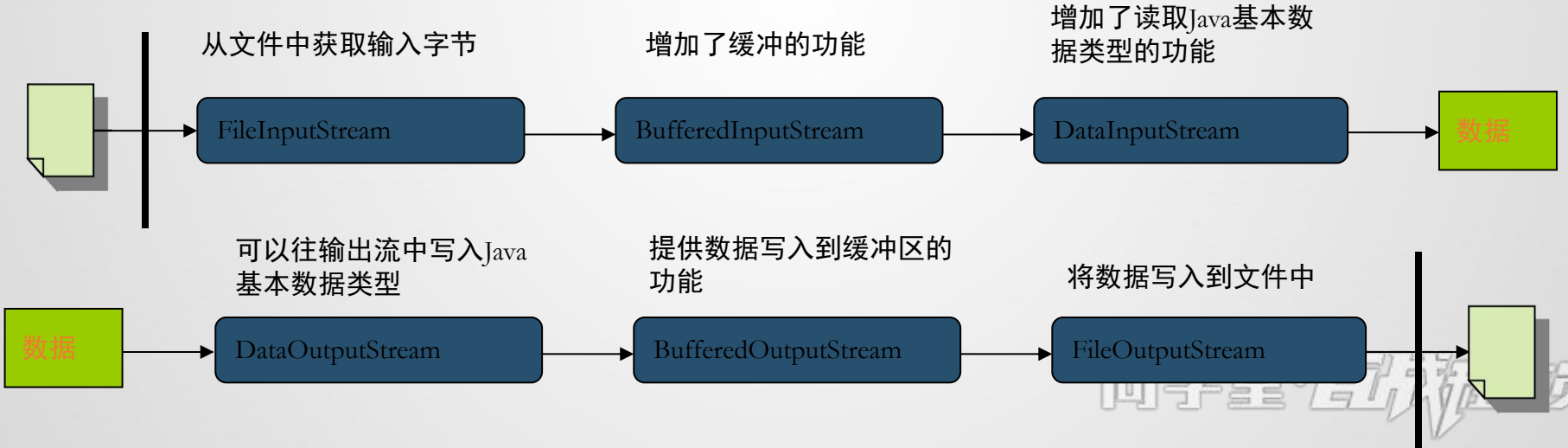
```
Reader reader = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(reader);
System.out.println("请输入用户名：");
String str = br.readLine();
System.out.println(str);
```

# 打印流

- PrintStream
  - PrintStream提供了一系列的print()和println()，可以实现将基本数据类型格式化成字符串输出。对象类型将先调用toString()，然后输出该方法返回的字符串
  - System.out就是PrintStream的一个实例，代表显示器
  - System.err 也是PrintStream的一个实例，代表显示器
  - PrintStream的输出功能非常强大，通常需要输出文本内容，都可以将输出流包装成PrintStream后进行输出
  - PrintStream的方法都不抛出IOException
- PrintWriter
  - PrintStream的对应字符流，功能相同，方法对应。
  - PrintWriter的方法也不抛出IOException
  - 复制文件时可以使用PrintWriter代替BufferedWriter完成，更简单

# 缓冲字节流

- DataInputStream和DataOutputStream
  - 提供了可以存取所有Java基础类型数据（如：int，double 等）和String的方法。
  - 处理流，只针对字节流，二进制文件
- 输入流链和输出流链
  - 注意：只要关闭上层流即可



# 对象序列化

- 对象序列化 (Serialization)
  - 将Java对象转换成字节序列（IO字节流）
  - 对象反序列化 (DeSerialization)
  - 从字节序列中恢复Java对象

- 为什么序列化
  - 序列化以后的对象可以保存到磁盘上，也可以在网络上传输，使得不同的计算机可以共享对象.（序列化的字节序列是平台无关的）

- 对象序列化的条件
- 只有实现了Serializable接口的类的对象才可以被序列化。Serializable接口中没有任何的方法，实现该接口的类不需要实现额外的方法。
- 如果对象的属性是对象，属性对应类也必须实现Serializable接口

# 对象序列化

- 如何实现序列化
  - 创建ObjectOutputStream对象
  - 调用writeObject()输出对象

```
OutputStream fos = new FileOutputStream(new File("d:/java6.txt"));
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(stu);
oos.close();
```

- 如何实现反序列化
  - 创建ObjectInputStream对象
  - 调用readObject()读取对象

```
InputStream fis = new FileInputStream(new File("d:/java6.txt"));
ObjectInputStream ois = new ObjectInputStream(fis);
Student stu = (Student)ois.readObject();
System.out.println(stu.getAge()+"  "+stu.getScore());
```

# 对象序列化

- 序列化能保存的元素
  - 只能保存对象的非静态成员变量
  - 不能保存任何成员方法和静态的成员变量
  - 不保存transient成员变量
  - 如果一个对象的成员变量是一个对象，这个对象的成员变量也会保存
  - 串行化保存的只是变量的值，对于变量的任何修饰符，都不能保存

- 使用对象流把一个对象写到文件时不仅保证该对象是序列化的，而且该对象的成员对象也必须是可序列化的。
- 如果一个可序列化的对象包含对某个不可序列化的对象的引用，那么整个序列化操作将会失败，并且会抛出一个NotSerializableException。我们可以将这个引用标记为transient，那么对象仍然可以序列化。


# 对象序列化

- 同一个对象多次序列化的处理
  - 所有保存到磁盘中的对象都有一个序列化编号
  - 序列化一个对象中，首先检查该对象是否已经序列化过
  - 如果没有，进行序列化
  - 如果已经序列化，将不再重新序列化，而是输出编号即可
- 如果不希望某些属性（敏感）序列化，或不希望出现递归序列
  - 为属性添加transient关键字（完成排除在序列化之外）
  - 自定义序列化（不仅可以决定哪些属性不参与序列化，还可以定义属性具体如何序列化）
- 序列化版本不兼容
  - 修改了实例属性后,会影响版本号，从而导致反序列化不成功
  - 解决方案：为Java对象指定序列化版本号serialVersionUID

# 字节/字符数组/字符串流

- ByteArrayInutStream和ByteArrayOutputStream
  - 数据源或目的地为：字节数组
  - 只有字节流，没有字符流
  - 节点流
- CharArrayReader和CharArrayWriter
  - 数据源或目的地为：字符数组
  - 只有字符流，没有字节流
  - 节点流
- StringReader和StringWriter
  - 数据源或目的地为：字符串
  - 只有字符流，没有字节流
  - 节点流

# 总结-Java IO体系

- IO基础
  - 流的原理
    - 对于数据的输入/输出操作以“流” (stream) 方式进行
    - 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
  - 流的分类
    - 输入流和输出流
    - 字节流和字符流
    - 节点流和处理流（包装流 装饰流）
  - 流的体系
    - InputStream 字节输入流和OutputStream 字节输出流
    - Reader 字符输入流和Writer 字符输出流

# 总结-JavaIO体系

- 具体IO介绍
  - 文件流：节点流
    - FileInputStream和FileOutputStream　FileReader和FileWriter
  - 缓冲流：包装流
    - BufferedInputStream和BufferedOutputStream　BufferedReader和BufferedWriter
  - 转换流：包装流 字节流转换成字符流 System.in
    - InputStreamReader和OutputStreamWriter
  - 打印流：包装流 只有输出流 System.out
    - PrintStream和PrintWriter
  - 数据流：包装流 只有字节流 基本类型和String
    - DataInputStream和DataOutputStream
  - 对象流：包装流 只有字节流 序列化 对象
    - ObjectInputStream和ObjectOutputStream
  - 字节数组流：节点流 字节流
    - ByteArrayInutStream和ByteArrayOutputStream

# 总结-JavaIO体系

| 分类 | 字节输入流 | 字节输出流 | 字符输入流 | 字符输出流 |
| --- | --- | --- | --- | --- |
| 抽象基类 | InputStream | OutputStream | Reader | Writer |
| 访问文件 | FileInputStream | FileOutputStream | FileReader | FileWriter |
| 访问数组 | ByteArray InputStream | ByteArray OutputStream | CharArray Reader | CharArray Writer |
| 访问管道 | Piped InputStream | Piped OutputStream | PipedReader | PipedWriter |
| 访问字符串 | | | StringReader | StringWriter |
| 缓冲流 | Buffered InputStream | Buffered OutputStream | Buffered Reader | Buffered Writer |
| 转换流 | | | InputStream Reader | OutputStream Writer |
| 对象流 | Object InputStream | Object OutputStream | | |
| 抽象基类 | Filter InputStream | Filter OutputStream | FilterReader | FilterWriter |
| 打印流 | | PrintStream | | PrintWriter |
| 推回输入流 | Pushback InputStream | | Pushback Reader | |
| 特殊流 | DataInputStream | DataOutputStream | | |

# 总结-Java IO体系

- 扩展内容
  - 其他IO流
  - RandomAccessFile类
  - NIO类
  - IO相关的设计模式
    - 装饰模式
      - 继承的一种替代方案
      - 如果采用继承，子类会很多，装饰模式就是各种IO流动态组装
    - 适配器模式　3相/2相　2.5/3.5　5V/3V
      - InputStreamReader和OutputStreamWriter

# 作业

- 文件的拷贝
- 利用递归实现将某个目录下所有内容copy到另一个目录中。
- 使用IO完成京东商城商品的添加和查询操作