# Java语言与系统设计

# 第11讲 输入输出流

- File类
- IO流原理
- 节点流（文件流）
- 缓冲流
- 转换流
- 标准输入输出流
- 打印流
- 数据流
- 对象流
- 随机存取文件流

# 1. File类

- java.io.File类：文件和文件目录路径的抽象表示形式，与平台无关
- File 能新建、删除、重命名文件和目录，但File 不能访问文件内容本身。如果需要访问文件内容本身，则需要使用输入/输出流。
- 想要在Java程序中表示一个真实存在的文件或目录，那么必须有一个File对象，但是Java程序中的一个File对象，可能没有一个真实存在的文件或目录。
- File对象可以作为参数传递给流的构造函数。

# 1. File类

- 构造函数

- public File(String pathname):以pathname为路径创建File对象，可以是绝对路径或者相对路径，如果pathname是相对路径，则默认的当前路径在系统属性user.dir中存储。
- public File(String parent, String child):以parent为父路径，child为子路径创建File对象。
- public File(File parent, String child):根据一个父File对象和子文件路径创建File对象

# 1. File类

- 路径分隔符
  - 路径中的每级目录之间用一个路径分隔符隔开。
  - windows和DOS系统默认使用“\”来表示
  - UNIX和URL使用“/”来表示
  - File类提供了一个常量：public static final String separator。根据操作系统，动态的提供分隔符。

例如：

```
File file1= newFile("c:\\test\\info.txt");
File file2= newFile("c:"+ File.separator+ "test"+ File.separator+ "info.txt");
File file3= newFile("c:/test");
```

# 1. File类

- File类的创建功能
  - public boolean createNewFile()：创建文件。若文件存在，则不创建，返回false
  - public boolean mkdir()：创建文件目录。如果此文件目录存在，就不创建了。如果此文件目录的上层目录不存在，也不创建。
  - public boolean mkdirs()：创建文件目录。如果上层文件目录不存在，一并创建

  注意事项：如果你创建文件或者文件目录没有写盘符路径，那么，默认在项目路径下。

# 1. File类

- File类的删除功能
  - public boolean delete()：删除文件或者文件夹
  - 注意事项：要删除一个文件目录，请注意该文件目录内不能包含文件或者文件目录

# 1. File类

- File类的方法
- public String getAbsolutePath()：获取绝对路径
- public String getPath()：获取路径
- public String getName()：获取名称
- public String getParent()：获取上层文件目录路径。若无，返回null
- public long length()：获取文件长度（即：字节数）。不能获取目录的长度。
- public long lastModified()：获取最后一次的修改时间，毫秒值

# 1. File类

```
File file1 = new File("D:\\名词总结.txt");
File file2 = new File("D:\\名词总结1.txt");
```


当硬盘中真有一个真实的文件或目录存在时，创建File对象时，各个属性会显式赋值。

当硬盘中没有真实的文件或目录对应时，那么创建对象时，除了指定的目录和路径之外，其他的属性都是取成员变量的默认值。

# 1. File类

- File类的方法
- public String[] list()：获取指定目录下的所有文件或者文件目录的名称数组
- public File[] listFiles()：获取指定目录下的所有文件或者文件目录的File数组
- public boolean renameTo(Filedest):把文件重命名为指定的文件路径

# 1. File类

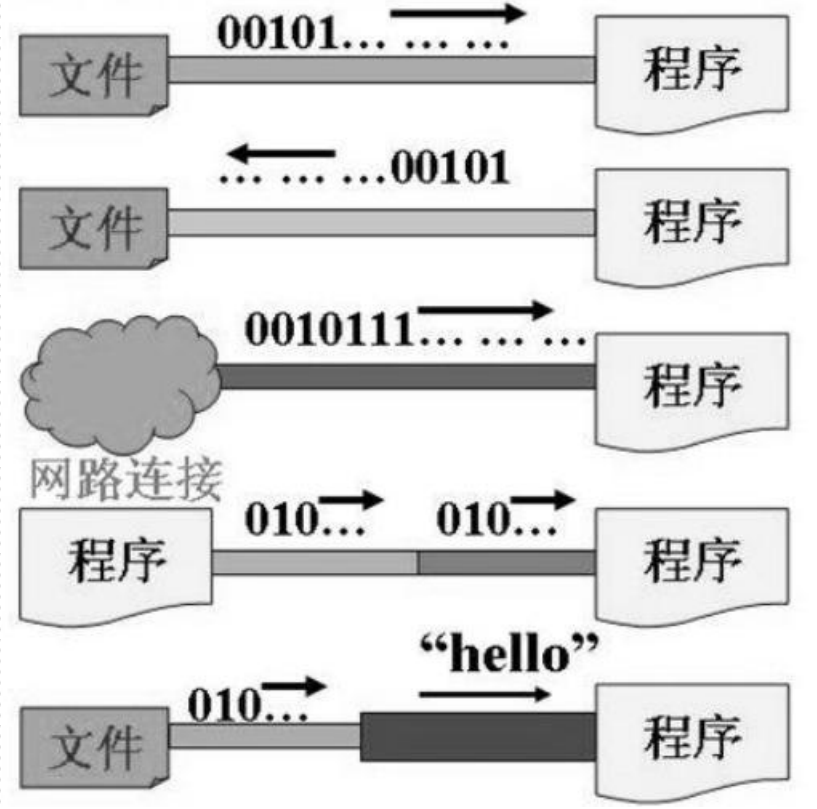
- File类的方法
- public boolean isDirectory()：判断是否是文件目录
- public boolean isFile()：判断是否是文件
- public boolean exists()：判断是否存在
- public boolean canRead()：判断是否可读
- public boolean canWrite()：判断是否可写
- public boolean isHidden()：判断是否隐藏

# 2. IO流的原理

- I/O是Input/Output的缩写，I/O技术是非常实用的技术，用于处理设备之间的数据传输。如读/写文件，网络通讯等。
- Java程序中，对于数据的输入/输出操作以“流(stream)”的方式进行。
- java.io包下提供了各种“流”类和接口，用以获取不同种类的数据，并通过标准的方法输入或输出数据。

# 2. IO流的原理

- 输入input：读取外部数据（磁盘、光盘等存储设备的数据）到程序（内存）中。
- 输出output：将程序（内存）数据输出到磁盘、光盘等存储设备中。



# 2. IO流的原理

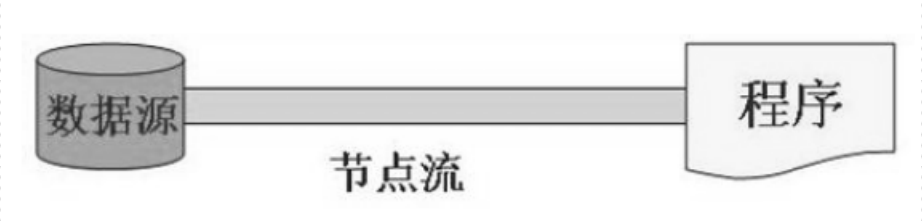
- 按操作数据单位不同分为：字节流(8 bit)，字符流(16 bit)
- 按数据流的流向不同分为：输入流，输出流
- 按流的角色的不同分为：节点流，处理流

| (抽象基类) | 字节流 | 字符流 |
| --- | --- | --- |
| 输入流 | InputStream | Reader |
| 输出流 | OutputStream | Writer |

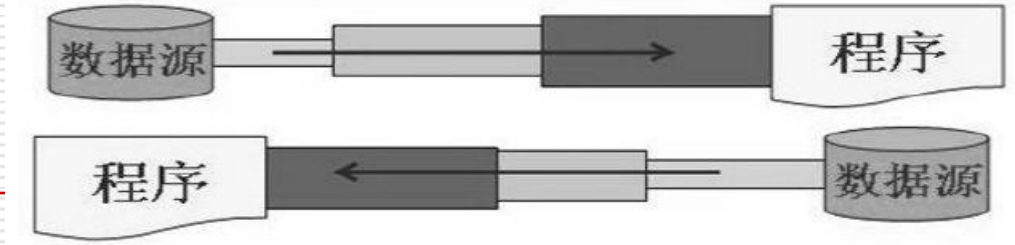
- Java的IO流共涉及40多个类，实际上非常规则，都是从4个抽象基类派生的。由这四个类派生出来的子类名称都是以其父类名作为子类名后缀。

# 2. IO流的原理

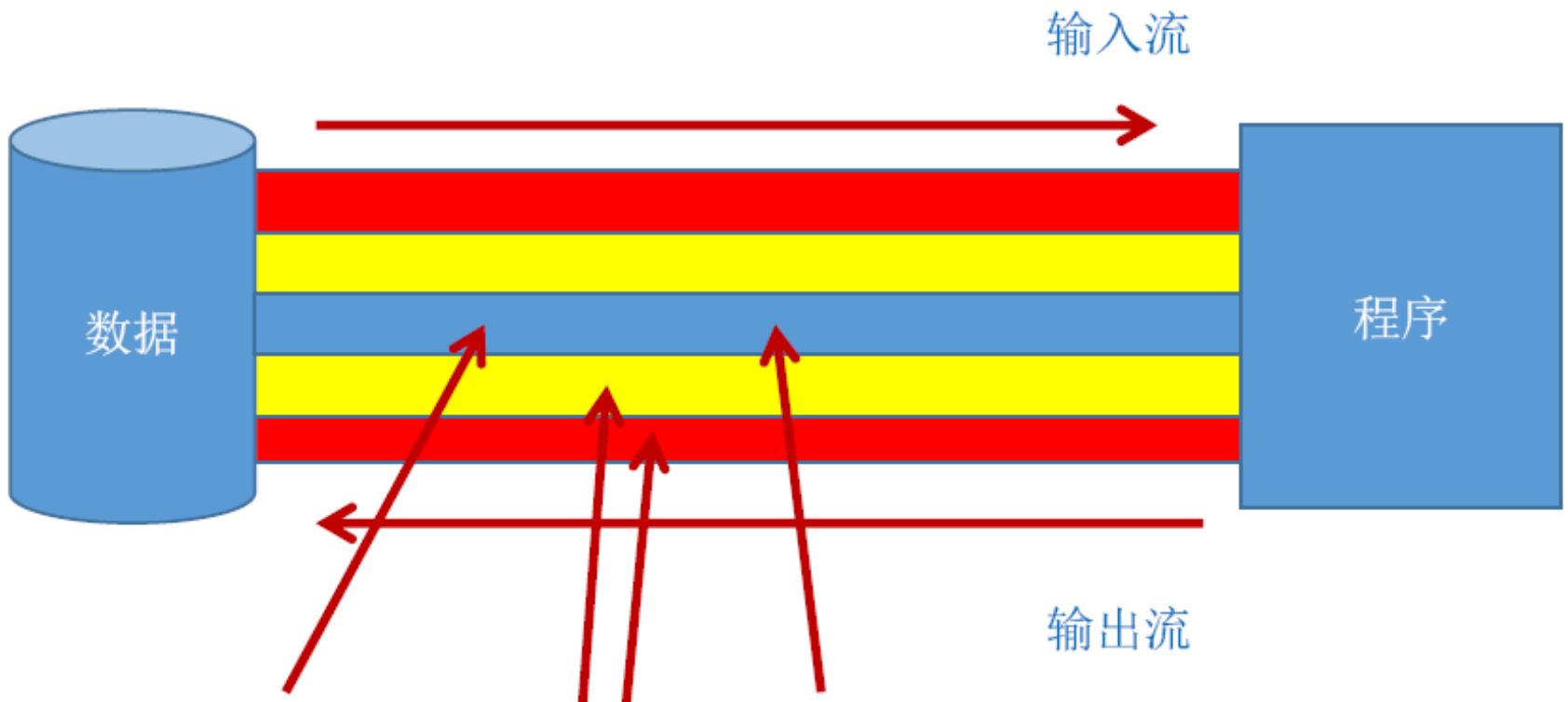
- 节点流：直接从数据源或目的地读写数据




- 处理流：不直接连接到数据源或目的地，而是“连接”在已存在的流（节点流或处理流）之上，通过对数据的处理为程序提供更为强大的读写功能。



# 2. IO流的原理




按照数据单位：
字节流
字符流

处理流

节点流：
FileInputStream/FileOutputSteam
FileReader/FileWriter

# 2. IO流的原理

| 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 | ByteArrayInputStream | ByteArrayOutputStream | CharArrayReader | CharArrayWriter |
| 访问管道 | PipedInputStream | PipedOutputStream | PipedReader | PipedWriter |
| 访问字符串 | | | StringReader | StringWriter |
| 缓冲流 | BufferedInputStream | BufferedOuputStream | BufferedReader | BufferedWriter |
| 转换流 | | | InputStreamReader | OutputStreamWriter |
| 对象流 | ObjectInputStream | ObjectOutputStream | | |
| | FilterInputStream | FilterOutputStream | FilterReader | FilterWriter |
| 打印流 | | PrintStream | | PrintWriter |
| 推回输入流 | PushbackInputStream | | PushbackReader | |
| 特殊流 | DataInputStream | DataOutputStream | | |

# 2. IO流的原理

- InputStream& Reader
- InputStream和Reader 是所有输入流的基类。
- InputStream（典型实现：FileInputStream）
- Reader（典型实现：FileReader）
- 程序中打开的文件IO 资源不属于内存里的资源，垃圾回收机制无法回收该资源，所以应该显式关闭文件IO 资源。
- FileInputStream 从文件系统中的某个文件中获得输入字节。FileInputStream 用于读取非文本数据之类的原始字节流。要读取字符流，需要使用FileReader.

# 2. IO流的原理

- InputStream
- int read():从输入流中读取数据的下一个字节。返回0到255范围内的int字节值。
- int read(byte[] b):从此输入流中将最多b.length个字节的数据读入一个byte数组中。
- int read(byte[] b,int off,int len): 将输入流中最多len个数据字节读入byte数组。
- public void close() throws IOException:关闭此输入流并释放与该流关联的所有系统资源。

# 2. IO流的原理

- Reader
- int read():读取单个字符。
- int read(char[] cbuf): 将字符读入数组。
- int read(char[] cbuf,int off,int len):将字符读入数组的某一部分。存到数组cbuf中，从off处开始存储
- public void close() throws IOException:关闭此输入流并释放与该流关联的所有系统资源。

# 2. IO流的原理

- OutputStream& Writer
- OutputStream 和Writer 也非常相似：
- 因为字符流直接以字符作为操作单位，所以Writer 可以用字符串来替换字符数组，即以String 对象作为参数
- FileOutputStream 从文件系统中的某个文件中获得输出字节。FileOutputStream 用于写出非文本数据之类的原始字节流。要写出字符流，需要使用FileWriter

# 2. IO流的原理

- OutputStream
- 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。
- public void flush() throws IOException:刷新此输出流并强制写出所有缓冲的输出字节
- public void close() throws IOException:关闭此输出流并释放与该流关联的所有系统资源。

# 2. IO流的原理

- Writer
- void write(int c):写入单个字符。
- void write(char[] cbuf):写入字符数组。
- void write(char[] cbuf, int off, int len):写入字符数组的某一部分。
- void write(String str):写入字符串。
- void write(String str, int off, int len): 写入字符串的某一部分。
- void flush():刷新该流的缓冲，则立即将它们写入预期目标。
- public void close() throws IOException:关闭此输出流并释放与该流关联的所有系统资源。

# 3. 节点流（文件流）

- 读取文件
  - 1.建立一个流对象，将已存在的一个文件加载进流。
  - 2.创建一个临时存放数据的数组或变量。
  - 3.调用流对象的读取方法将流中的数据读入到数组或变量中。
  - 4. 关闭资源。
- 写入文件
  - 1.创建流对象，建立数据存放文件
  - 2.调用流对象的写入方法，将数据写入流
  - 3.关闭流资源，并将流中的数据清空到文件中。

# 文件读写的例子

```
import java.io.*;
public class CopyFile {
    public static void main(String[] args) {
        int dt;
        FileInputStream fis;
        FileOutputStream fos;
        try{ fis = new FileInputStream("c:\\file\\file1.doc");
        }catch(FileNotFoundException e){
                System.out.println("源文件未找到");
                return;
        }
        try{ fos = new FileOutputStream("c:\\file\\file2.doc");
        }catch(FileNotFoundException e){
                System.out.println("目标文件打开失败");
                return;
        }
```

```
try{   while((dt=fis.read())!=-1)
            fos.write(dt);
            }catch(IOException e){
                  System.out.println("文件读写出错");
            }finally{
            try{
            fis.close();
            fos.close();
            }
            catch(IOException e){e.printStackTrace();}
            finally{System.out.println("文件读写完毕");}
            }
    }
}
```

# 3. 节点流（文件流）

- 在写入一个文件时，如果使用构造器File OutputStream(file)，则目录下有同名文件将被覆盖。
- 如果使用构造器File OutputStream(file,true)，则目录下的同名文件不会被覆盖，在文件内容末尾追加内容。
- 在读取文件时，必须保证该文件已存在，否则报异常。
- 字节流操作字节，比如：.mp3，.avi，.rmvb，mp4，.jpg，.doc，.ppt
- 字符流操作字符，只能操作普通文本文件。最常见的文本文件：.txt，.java，.c，.cpp 等语言的源代码。尤其注意 .doc,excel,ppt这些不是文本文件。

# 4. 缓冲流

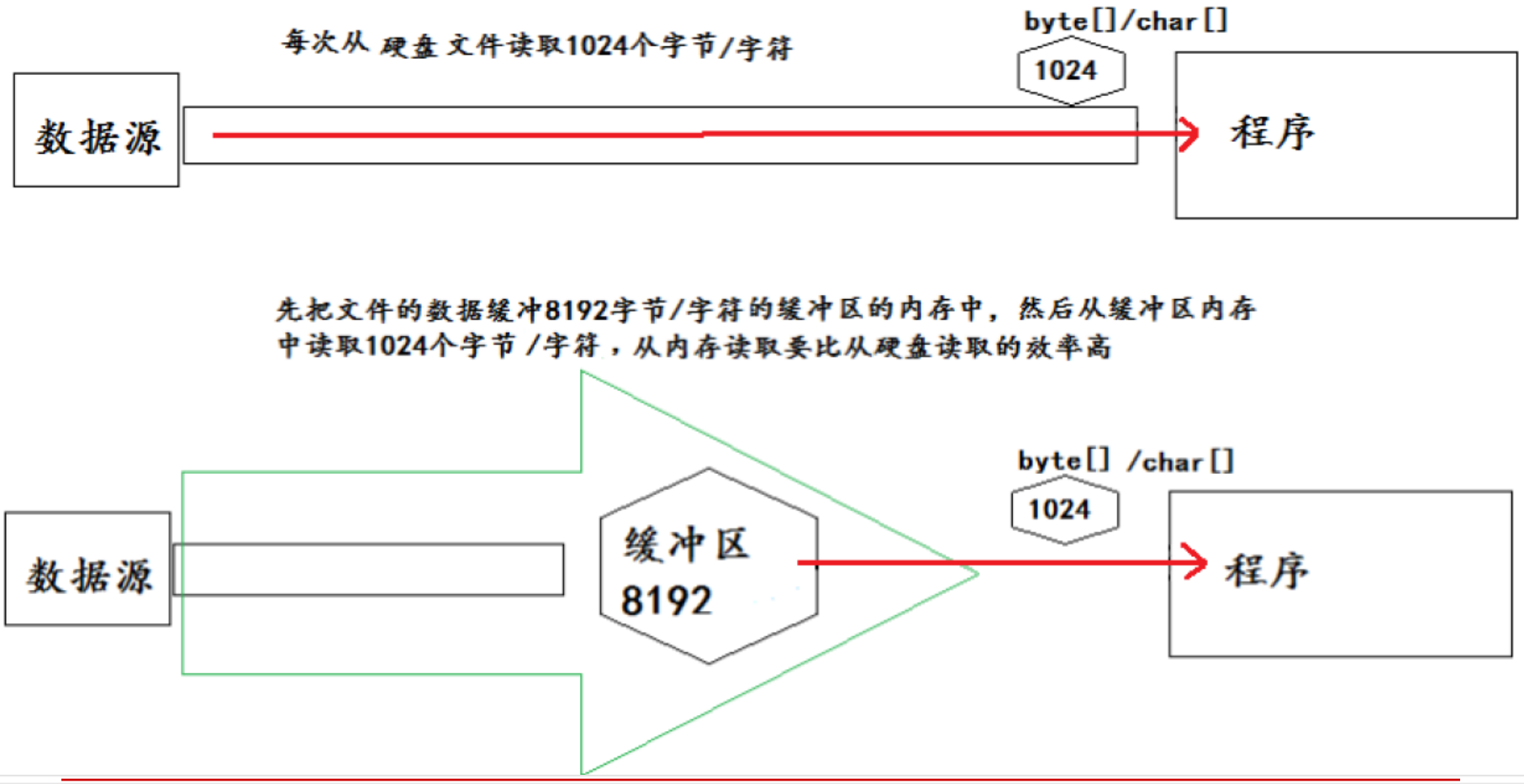
- 为了提高数据读写的速度，Java API提供了带缓冲功能的流类，在使用这些流类时，会创建一个内部缓冲区数组，缺省使用8192个字节(8Kb)的缓冲区。

```
public
class BufferedInputStream extends FilterInputStream {

    private static int DEFAULT_BUFFER_SIZE = 8192;
```

- 缓冲流要“套接”在相应的节点流之上，根据数据操作单位可以把缓冲流分为：
  - BufferedInputStream和BufferedOutputStream
  - BufferedReader和BufferedWriter

# 4. 缓冲流


每次从 硬盘 文件读取1024个字节/字符
byte[]/char[]
1024
数据源
程序
先把文件的数据缓冲8192字节/字符的缓冲区的内存中，然后从缓冲区内存中读取1024个字节 /字符，从内存读取要比从硬盘读取的效率高
byte[] /char[]
1024
数据源
缓冲区
8192
程序

# 4. 缓冲流

- 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
- 当使用BufferedInputStream读取字节文件时，BufferedInputStream会一次性从文件中读取8192个字节(8Kb)，存在缓冲区中，直到缓冲区装满了，才重新从文件中读取下一个8192个字节数组。
- 向流中写入字节时，不会直接写到文件，先写到缓冲区中直到缓冲区写满，BufferedOutputStream才会把缓冲区中的数据一次性写到文件里。使用方法flush()可以强制将缓冲区的内容全部写入输出流

# 4. 缓冲流

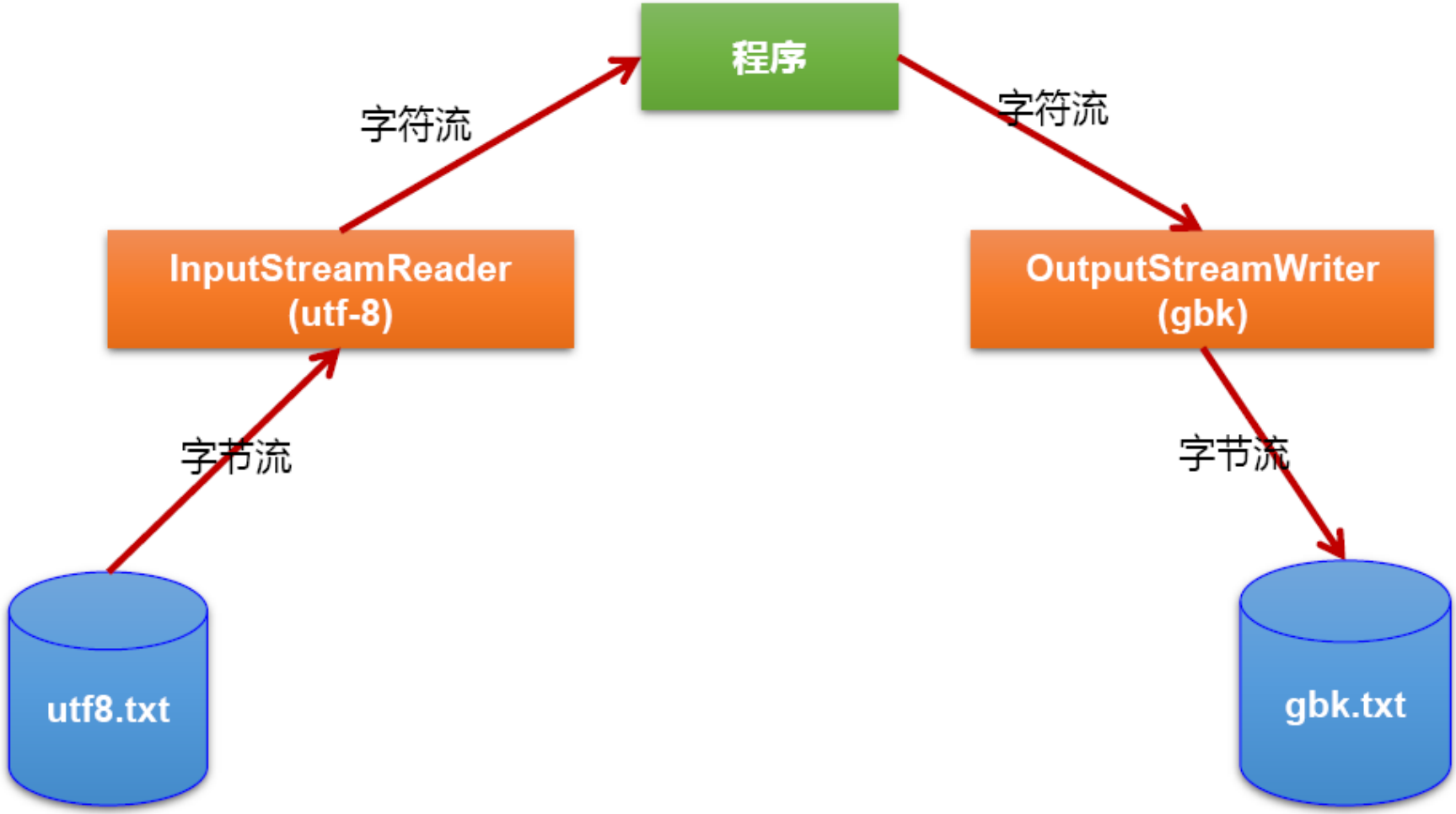
- 关闭流的顺序和打开流的顺序相反。只要关闭最外层流即可，关闭最外层流也会相应关闭内层节点流
- flush()方法的使用：手动将buffer中内容写入文件
- 如果是带缓冲区的流对象的close()方法，不但会关闭流，还会在关闭流之前刷新缓冲区

```
import java.io.*;
public class BufferReaderTest {
   public static void main(String[] args) {
try{FileReader in = new FileReader("c:\\file\\file1.txt");
              BufferedReader br = new BufferedReader(in);
              String line;
              while((line = br.readLine()) != null){
                     System.out.print(line);
             //读入数据时不包括行结束符，显示时补充换行显示
                     System.out.println();
              }
              in.close();
       }catch(IOException e){
              System.out.println(e.toString());
       }
   }
```

# 5. 转换流

- 转换流提供了在字节流和字符流之间的转换
- InputStreamReader：将InputStream转换为Reader
- OutputStreamWriter：将Writer转换为OutputStream
- 字节流中的数据都是字符时，转成字符流操作更高效。
- 很多时候我们使用转换流来处理文件乱码问题。实现编码和解码的功能。
- 解码：字节-》字符
- 编码：字符-》字节

# 5. 转换流


程序
字符流
字符流
InputStreamReader
(utf-8)
OutputStreamWriter
(gbk)
字节流
字节流
utf8.txt
gbk.txt

# 5. 转换流

- InputStreamReader
- 实现将字节的输入流按指定字符集转换为字符的输入流。
- 需要和InputStream“套接”。
- 构造函数:
- public InputStreamReader(InputStream in)
- public InputSreamReader(InputStream in,String charsetName)
- 例如：Reader isr= new InputStreamReader(System.in,”gbk”);

# 5. 转换流

- OutputStreamWriter
  - 实现将字符的输出流按指定字符集转换为字节的输出流。
  - 需要和OutputStream“套接”。
  - 构造函数：
  - public OutputStreamWriter(OutputStream out)
  - public OutputSreamWriter(OutputStream out, String charsetName)

# 6. 标准输入输出流

- System.in和System.out分别代表了系统标准的输入和输出设备
- 默认输入设备是：键盘，输出设备是：显示器
- System.in的类型是InputStream
- System.out的类型是PrintStream，其是OutputStream的子类

# 将键盘输入字符输出到屏幕并统计输入的字符数

```
import java.util.Scanner;
class MyIO {
    public static void main(String []args) {
        int count = 0;
        String str = "";
        Scanner rd = new Scanner(System.in);//JDK5提供Scanner
        System.out.println("请输入数据: ");
        while(rd.hasNext()) {
            str = rd.next();     //读入字符串数据
            System.out.print(str);      //显示刚输入字符串
            count += str.length();      //统计以读入的字符数目
        }
        System.out.println("\n共输入了" + count + "个字符");
    }
}
```

# 7. 打印流

- 实现将基本数据类型的数据格式转化为字符串输出
- 打印流：PrintStream和PrintWriter
- 提供了一系列重载的print()和println()方法，用于多种数据类型的输出
- PrintStream 打印的所有字符都使用平台的默认字符编码转换为字节。在需要写入字符而不是写入字节的情况下，应该使用 PrintWriter 类。
- System.out返回的是PrintStream的实例

# 打印九九乘法表

```
import java.io.*;
public class printTest {
       public static void main(String[] args) {
try{OutputStream os = new FileOutputStream("c:\\file\\data.txt");
       PrintStream ps = new PrintStream(os);
       for(int i=1; i<=9; i++){
          for(int j=1; j<=i; j++){
              ps.printf("%8s",i+"*"+j+"="+(i*j));}
              ps.println();
       }
       ps.close();
       os.close();
       }catch(IOException e){System.out.println(e.toString());}
       }
}
```

# 8. 数据流

- 为了方便地操作Java语言的基本数据类型和String的数据，可以使用数据流。
  - 数据流有两个类：(用于读取和写出基本数据类型、String类的数据)
  - DataInputStream和DataOutputStream分别“套接”在InputStream和OutputStream子类的流上
- 例如，DataInputStream中的方法

```
boolean readBoolean()
char readChar()
double readDouble()
long readLong()
String readUTF()
byte readByte()
float readFloat()
short readShort()
int readInt()
void readFully(byte[] b)
```

# 9. 对象流

- ObjectInputStream和OjbectOutputSteam
- 用于存储和读取基本数据类型数据或对象的处理流。它的强大之处就是可以把Java中的对象写入到数据源中，也能把对象从数据源中还原回来。

- 序列化：用ObjectOutputStream类保存基本类型数据或对象的机制
- 反序列化：用ObjectInputStream类读取基本类型数据或对象的机制

# 9. 对象流

- 序列化：将对象写入到磁盘或者进行网络传输

```
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(“data.txt"));
Person p = new Person("韩梅梅", 18, "中华大街", new Pet());
oos.writeObject(p);
oos.flush();
oos.close();
```

- 反序列化：将磁盘中的对象数据源读出。

```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream(“data.txt"));
Person p1 = (Person)ois.readObject();
System.out.println(p1.toString());
ois.close();
```

# 10. 随机存取文件流

- RandomAccessFile 声明在java.io包下，但直接继承于java.lang.Object类。并且它实现了DataInput、DataOutput这两个接口，也就意味着这个类既可以读也可以写。
- RandomAccessFile 类支持“随机访问” 的方式，程序可以直接跳到文件的任意地方来读、写文件
- RandomAccessFile 对象包含一个记录指针，用以标示当前读写处的位置。
- RandomAccessFile类对象可以自由移动记录指针：
  - long getFilePointer()：获取文件记录指针的当前位置
  - void seek(long pos)：将文件记录指针定位到pos位置

# 10. 随机存取文件流

- 构造函数：
  - public RandomAccessFile(File file, String mode)
  - public RandomAccessFile(String name, String mode)

- mode 参数，该参数指定RandomAccessFile的访问模式：
  - r: 以只读方式打开
  - rw：打开以便读取和写入

```
import java.io.*;
public class RandFileTest{
   public static void main(String[] args){
       String sFile="c:\\file\\data.txt";
       try{       //构造随机访问文件，使用可读写方式。
      RandomAccessFile rf = new  RandomAccessFile(sFile, "rw");
   for(int i = 0; i < 10; i++){//向文件中写入10个double类型的数据
               rf.writeDouble(i*100);}
               rf.close();
               //构造一个随机访问文件，使用只写方式
               rf = new RandomAccessFile(sFile, "rw");
               //移动文件指针到写入的第6个数据位置，重新写入数据100
               rf.seek(5*8);
               rf.writeDouble(100);
               rf.close();
```

```
    //构造一个随机文件访问文件，使用只读方式
            rf = new RandomAccessFile(sFile, "r");
            for(int i = 0; i < 10; i++){
    System.out.println("Value " + i + ": " + rf.readDouble());
            }
            rf.close();
    }catch(IOException e){
            System.out.println(e);
    }}}
}
```

# 习题

Java中有几种类型的流？JDK为每种类型的流提供了一些抽象类以供继承，请指出它们分别是哪些类?

【答案】Java中按所操作的数据单元的不同，分为字节流和字符流。

字节流继承于InputStream和OutputStream类;

字符流继承于Reader和Writer。

按流的流向的不同，分为输入流和输出流。

按流的角色来分，可分为节点流和处理流。缓冲流、转换流、对象流和打印流等都属于处理流，使得输入/输出更简单，执行效率更高。