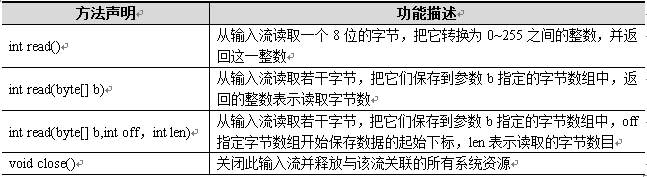
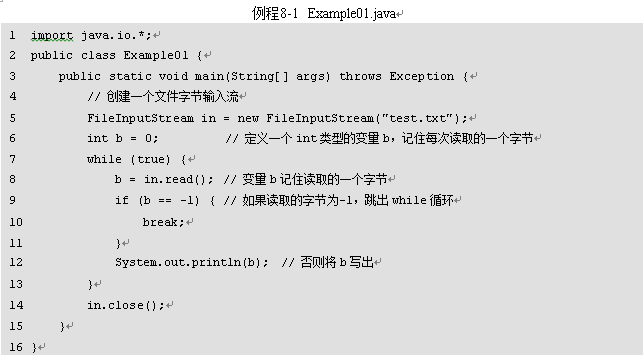
### InputStream

InputStream是所有字节输入流顶级父类

InputStream：

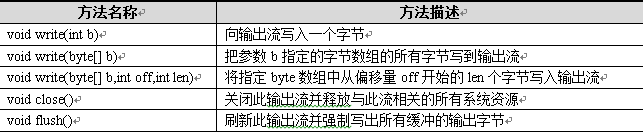




### 2.OutputStream

OutputStream是所有字节输出流顶级父类

OutputStream：



案例：

### 3.尝试：

/\*\*

\* 输入输出流

\*/

public class FileUtil {

/\*\*

\* 写操作

\* @param str

\*/

public static void fileOutput(String content,String fileName){

FileOutputStream stream = null;

try {

stream = new FileOutputStream(fileName);

byte[] bytes = content.getBytes();

for (int i = 0; i < bytes.length; i++) {

stream.write(bytes[i]);

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally{

try {

stream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

/\*\*

\* 读操作

\* @param fileName

\*/

public static void fileInput(String fileName) {

File file = new File(fileName);

//1、创建输入流

FileInputStream stream = null;

try {

if (!file.exists()) {

file.createNewFile();

}

stream = new FileInputStream(file);

int length = 0;//保存读取的长度

byte[] bytes = new byte[1024];//保存读到的内容

//开始读操作

while ((length = stream.read(bytes)) != -1) {//读取一个字节

System.out.write(bytes,0,length);

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally{

try {

stream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

public static void main(String[] args) {

String str = "学习真是很漫长的过，才第三天呀，还有12天，天啦";

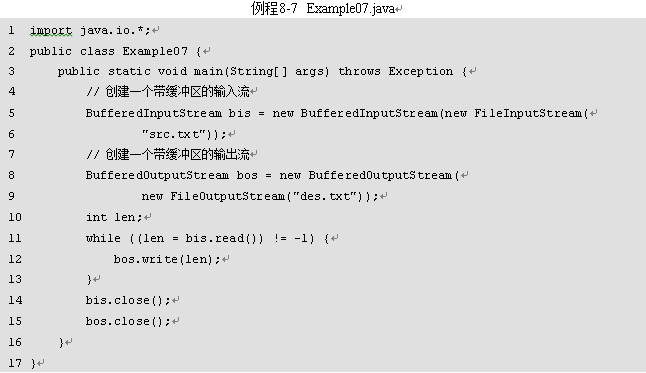
String name = "ddup";

FileUtil.fileOutput(str,"ddup");

FileUtil.fileInput(name);

}

### 4.过滤流：



### 总结：

顶层：InputStream、OutPutstream

直接子类：FileInputStream、FileOutPutstream

包装类：BufferedInputStream、BufferedOutPutstream：首选

### 字符流

顶层：Reader、Writer

子类FileReader、FileWriter

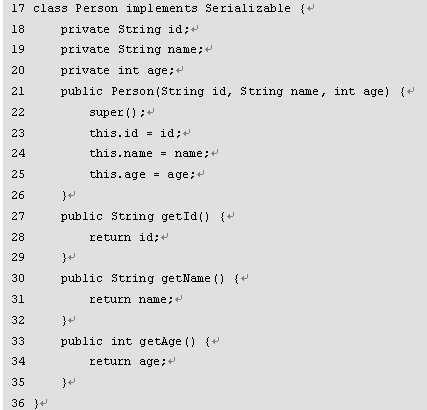
包装类：BufferedReader、BufferedWriter

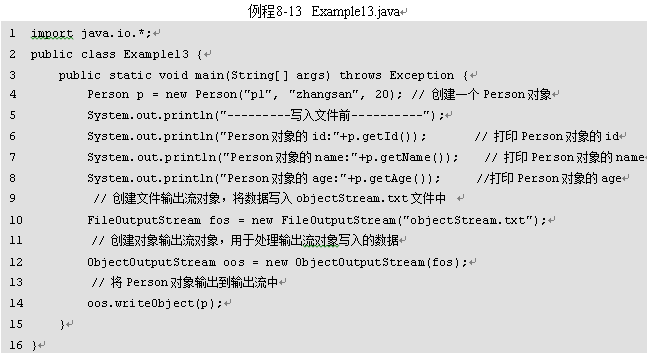
次包装类：LineNumberReader、LineNumberWriter

### 转换流

OutputStreamWriter、InputStreamReader

### 对象流





### File

Root

Tree1 tree2

First.txt second.txt

举例：

public static void create() {

//創建文件甲

File root = new File("root");

root.mkdir();

File tree1 = new File(root,"tree1");

tree1.mkdir();

File first = new File(tree1,"first");

try {

first.createNewFile();

} catch (IOException e) {

e.printStackTrace();

}

}

### 11.文件操作：

* + String getName( )； //得到一个文件的名称（不包括路径）
  + String getPath( )； //得到一个文件的路径名
  + String getAbsolutePath( )；//得到一个文件的绝对路径名
  + String getParent( )； //得到一个文件的上一级目录名
  + File getParentFile()；//返回父文件对象
  + 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( )； //测试当前File对象所指示的文件是否存在
  + boolean canWrite( )；//测试当前文件是否可写
  + boolean canRead( )；//测试当前文件是否可读
  + Boolean isHidden(); //判断文件是否是隐藏的
  + boolean isFile( )； //测试当前文件是否是文件（不是目录）
  + boolean isDirectory( )； //测试当前文件是否是目录
  + 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
  + 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
  + boolean mkdir( ); //根据当前对象生成一个由该对象指定的路径
  + String[ ] list( ); //列出当前目录下的文件
  + File[] listFiles();//返回当前目录下的所有文件对象