* **处理流之一：缓冲流（可以提升文件的传输效率）**
* 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**。
* 根据数据操作单位可以把缓冲流分为：
  + **BufferedInputStream和BufferedOutputStream**
  + **BufferedReader和BufferedWriter**
* 缓冲流要“套接”在相应的节点流之上，对读写的数据提供了缓冲的功能，提高了读写的效率，同时增加了一些新的方法。
* 对于输出的缓冲流，写出的数据会先在内存中缓存，使用**flush()**将会使内存中的数据立刻写出。
* **使用BufferedInputStream和BufferedOutputStream实现非文本文件的复制。**
* 步骤：创建BufferedInputStream和BufferedOutputStream缓冲流对象；
* 提供读入、写出的文件File的对象；
* 创建相应的节点流FileInputStream和FileOutputStream的对象；
* 将创建的节点流的对象作为形参传递给缓冲流的构造器中；
* 实现文件复制的操作；
* 关闭缓冲流。

@Test  
public void test1() {  
 BufferedInputStream bis = null;  
 BufferedOutputStream bos = null;  
  
 try {  
 //1）提供读入、写出的文件。  
 File file1 = new File("D:\\io\\1.gif");  
 File file2 = new File("D:\\io\\2.gif");  
  
 //2）创建相应的节点流：FileInputStream、FileOutputStream。  
 FileInputStream fis = new FileInputStream(file1);  
 FileOutputStream fos = new FileOutputStream(file2);  
  
 //3）将创建的节点流的对象作为形参传递给缓冲流的构造器中。  
 bis = new BufferedInputStream(fis);  
 bos = new BufferedOutputStream(fos);  
  
 //4）具体的实现文件复制的操作。  
 byte[] b = new byte[1024];  
 int len;  
 while ((len = bis.read(b)) != -1) {  
 bos.write(b, 0, len);  
 bos.flush();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 //5）关闭缓冲流。  
 if (bos != null) {  
 try {  
 bos.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 if (bis != null) {  
 try {  
 bis.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

* **写成复制方法**

@Test  
 public void test2() {  
 long start = System.currentTimeMillis();  
 String src = "D:\\io\\1.wmv";  
 String dest = "D:\\io\\2.wmv";  
// String src = "C:\\Users\\shkstart\\Desktop\\实验.doc";  
// String dest = "C:\\Users\\shkstart\\Desktop\\实验1.doc";  
 copyFile(src, dest);  
 long end = System.currentTimeMillis();  
 System.***out***.println("花费的时间为：" + (end - start));  
 }  
  
 /\*\*  
 \* 使用缓冲流实现文件的复制的方法  
 \*/  
 public void copyFile(String src, String dest) {  
 BufferedInputStream bis = null;  
 BufferedOutputStream bos = null;  
 try {  
 //1.提供读入、写出的文件  
 File file1 = new File(src);  
 File file2 = new File(dest);  
 //2.想创建相应的节点流：FileInputStream、FileOutputStream  
 FileInputStream fis = new FileInputStream(file1);  
 FileOutputStream fos = new FileOutputStream(file2);  
 //3.将创建的节点流的对象作为形参传递给缓冲流的构造器中  
 bis = new BufferedInputStream(fis);  
 bos = new BufferedOutputStream(fos);  
 //4.具体的实现文件复制的操作  
 byte[] b = new byte[1024];  
 int len;  
 while ((len = bis.read(b)) != -1) {  
 bos.write(b, 0, len);  
 bos.flush();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 //5.关闭相应的流  
 if (bos != null) {  
 try {  
 bos.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 if (bis != null) {  
 try {  
 bis.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }

* **readLine()的使用**

@Test  
public void test3() {  
  
 BufferedReader br = null;  
 BufferedWriter bw = null;  
  
 try {  
 File file1 = new File("D:\\io\\hello.txt");  
 File file2 = new File("D:\\io\\hello2.txt");  
  
 FileReader fr = new FileReader(file1);  
 FileWriter fw = new FileWriter(file2);  
  
 br = new BufferedReader(fr);  
 bw = new BufferedWriter(fw);  
  
 String str;  
 while ((str = br.readLine()) != null) {  
 bw.write(str + "\n");

//或者使用bw.newLine();来换行

bw.flush();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 if (br != null) {  
 try {  
 br.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 if (bw != null) {  
 try {  
 bw.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}