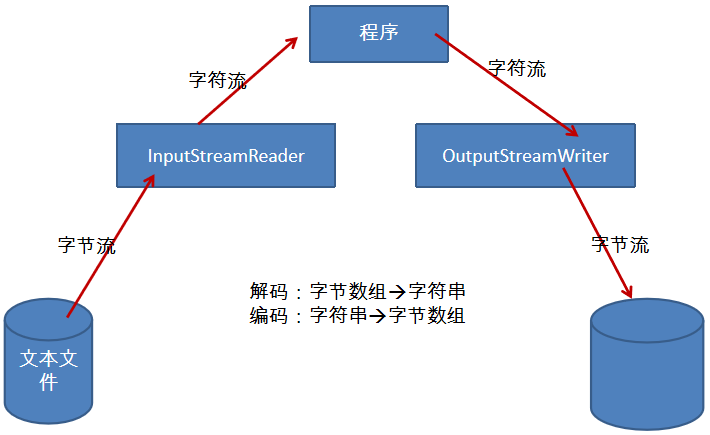
* **处理流之二：转换流**
* 转换流提供了在字节流和字符流之间的转换。
* Java API提供了两个转换流：InputStreamReader和OutputStreamWriter。
* 字节流中的数据都是字符时，转成字符流操作更高效。
* **InputStreamReader**
* 用于将字节流中读取到的字节按指定字符集解码成字符。需要和InputStream“套接”。
* 构造方法
  + public InputStreamReader(InputStream in)
  + public InputStreamReader(InputStream in,String charsetName)
* **OutputStreamWriter**
* 用于将要写入到字节流中的字符按指定字符集编码成字节。需要和OutputStream“套接”。
* 构造方法
* public OutputStreamWriter(OutputStream out)
* public OutputStreamWriter(OutputStream out,String charsetName)
* 如：Reader isr=new InputStreamReader(System.in,”ISO5334\_1”);



* **字符编码：**
* 编码表的由来：
* 计算机只能识别二进制数据，早期由来是电信号。为了方便应用计算机，让它可以识别各个国家的文字。就将各个国家的文字用数字来表示，并一一对应，形成一张表。这就是编码表。
* 常见的编码表：
  + **ASCII**：美国标准信息交换码。
* 用一个字节的7位可以表示。
  + **ISO8859-1**：拉丁码表。欧洲码表
* 用一个字节的8位表示。
  + **GB2312**：中国的中文编码表。
* GBK：中国的中文编码表升级，融合了更多的中文文字符号。
  + **Unicode**：国际标准码，融合了多种文字。
* 所有文字都用两个字节来表示,Java语言使用的就是unicode
  + **UTF-8**：最多用三个字节来表示一个字符。
  + 编码：字符串——>字节数组
  + 解码：字节数组——>字符串
  + 转换流的编码应用：
* 可以将字符按指定编码格式存储。
* 可以对文本数据按指定编码格式来解读。
* 指定编码表的动作由构造器完成。
* **实现字节流与字符流之间的转换**

@Test  
public void test1() {  
  
 BufferedReader br = null;  
 BufferedWriter bw = null;  
  
 try {  
 //解码  
 File file = new File("dbcp.txt");  
 FileInputStream fis = new FileInputStream(file);  
 InputStreamReader isr = new InputStreamReader(fis, "UTF-8");  
 br = new BufferedReader(isr);  
  
 //编码  
 File file1 = new File("dbcp4.txt");  
 FileOutputStream fos = new FileOutputStream(file1);  
 OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");  
 bw = new BufferedWriter(osw);  
  
 String str;  
 while ((str = br.readLine()) != null) {  
 bw.write(str);  
 bw.newLine();  
 bw.flush();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 if (bw != null) {  
 try {  
 bw.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 if (br != null) {  
 try {  
 br.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}