1、缓冲流

1. BufferedInputStream
2. BufferedOutputStream
3. BufferedReader
4. BufferedWriter

2、作用

提供流的读取、写入的速度

3、说明

1. 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;
```

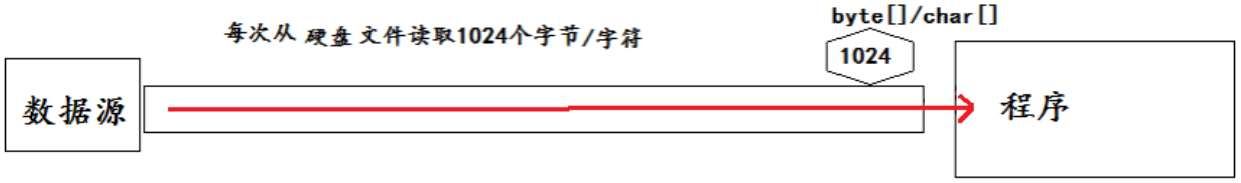
2. 缓冲 流要“套接”在相应的节点流之上，根据数据操作单位可以把缓冲流分为：
    1. BufferedInputStream 和 BufferedOutputStream（字节流）
    2. BufferedReader 和 BufferedWriter（字符流）
3. 当读取数据时，数据按块读入缓冲区，其后的读操作则直接访问缓冲区
4. 当使用 BufferedInputStream 读取字节文件时，BufferedInputStream 会一次性从文件中读取 8192 个 (8Kb 存在 缓冲区中直到缓冲区装满 了，才重新从文件中读取下一个 8192 个字节数组 。

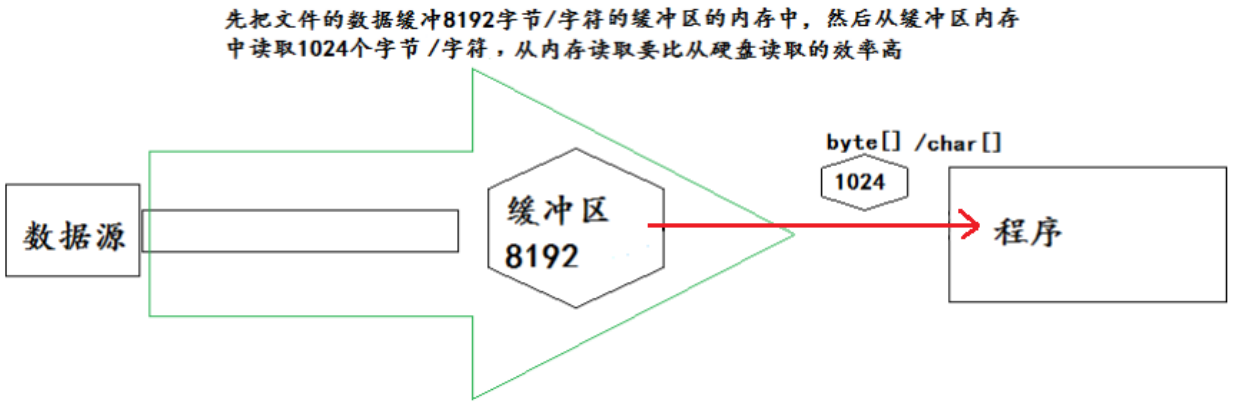
5. 向流中写入字节时，不会直接写到文件，先写到缓冲区中直到缓冲区写满，BufferedOutputStream 才会把缓冲区中的数据一次性写到文件里 。使用方法flush() 可以强制将缓冲区的内容全部写入输出流

6. 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

7. flush() 方法的使用：手动将 buffer 中内容写入文件

8. 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， 关闭后不能再写出





4、代码实例

```
1  public class BufferDemo {
2   public static void main(String[] args) {
3   // 非文本文件路径
4   String srcPath1 = "C:\\Users\\Administrator\\Desktop\\比特寒假班
\\JAVA\\FileExer1030\\1.avi";
5   String destPath1 = "C:\\Users\\Administrator\\Desktop\\比特寒假
班\\JAVA\\FileExer1030\\2.avi";
6
7   // 文本文件路径
8   String srcPath2 = "C:\\Users\\Administrator\\Desktop\\比特寒假班
\\JAVA\\FileExer1030\\test5.txt";
9   String destPath2 = "C:\\Users\\Administrator\\Desktop\\比特寒假
班\\JAVA\\FileExer1030\\test6.txt";
10
```

```
11  long start1 = System.currentTimeMillis();
12  // 不同路径下非文本文件的复制
13  copyFile1(srcPath1,destPath1);
14  long end1 = System.currentTimeMillis();
15  System.out.println("非文本文件复制花费的时间为: " + (end1 - start
1));
16
17  long start2 = System.currentTimeMillis();
18  // 不同路径下文本文件的复制
19  copyFile2(srcPath2,destPath2);
20  long end2 = System.currentTimeMillis();
21  System.out.println("非文本文件复制花费的时间为: " + (end2 - start
2));
22  }
23
24  // 不同路径下非文本文件的复制
25  private static void copyFile1(String srcPath, String destPath)
{
26  // 1.造文件
27  File srcFile = new File(srcPath);
28  File destFile = new File(destPath);
29
30  // 2.造流
31  // 2.1 造节点流
32  try (InputStream is = new FileInputStream(srcFile);
33  OutputStream os = new FileOutputStream(destFile)) {
34  // 2.2 造缓冲流
35  try (BufferedInputStream bis = new BufferedInputStream(is);
36  BufferedOutputStream bos = new BufferedOutputStream(os)) {
37  // 3.复制的细节: 读取、写入
38  int len;
39  byte[] cubf = new byte[1024];
40  while((len = bis.read(cubf)) != -1){
41  bos.write(cubf,0,len);
42  }
43  bos.flush();
```

```
44  }catch (Exception e){
45  e.printStackTrace();
46  }
47  }catch (Exception e){
48  e.printStackTrace();
49  }
50  }
51  private static void copyFile2(String srcPath, String destPath)
{
52  // 1.造文件
53  File srcFile = new File(srcPath);
54  File destFile = new File(destPath);
55
56  // 2.造流
57  // 2.1 造节点流
58  try (FileReader fr = new FileReader(srcFile);
59  FileWriter fw = new FileWriter(destFile)) {
60  // 2.2 造缓冲流
61  try (BufferedReader br = new BufferedReader(fr);
62  BufferedWriter bw = new BufferedWriter(fw)) {
63  // 3.复制的细节：读取、写入
64  int len;
65  char[] cubf = new char[1024];
66  while((len = br.read(cubf)) != -1){
67  bw.write(cubf,0,len);
68  }
69  bw.flush();
70  }catch (Exception e){
71  e.printStackTrace();
72  }
73  }catch (Exception e){
74  e.printStackTrace();
75  }
76  }
77 }
```