FileInputStream：

构造方法：

1.public FileInputStream(File file)

2.public FileInputStream(String name)

成员方法：  
1.public int available() //还有多少字节未读 内部调用本地方法

1. public int read() //读一个字节启动一次IO
2. public int read(byte b[], int off, int len) //读多个字节启动一次IO
3. public long skip(long n) //跳过n个字节不读
4. public synchronized void mark(int readlimit) //标记当前位置，在调用mark之后如果读取超过readlimit个字节后标记失效。FileInputStream的mark方法默认是空的
5. public boolean markSupported() //当前类是否支持mark和reset方法。FIS的这个方法返回false，默认不支持
6. public synchronized void reset() //返回到mark标记的位置 如果超过了readlimit的限制或者没有调用mark方法则抛出异常。FIS不支持这个方法，默认抛出异常

FileOutputStream：

构造方法：

1. public FileOutputStream(File file, boolean append) //append是否追加
2. public FileOutputStream(String name, boolean append)

成员方法：  
1.public void flush() //默认空实现 因为没有缓冲区

2.public void write(byte b[], int off, int len)

3.public void write(int b)

BufferedInputStream：继承自FilterInputStream

属性：

1.protected volatile byte buf[]; //缓冲

2.protected int count; //缓冲最后一个字节的位置

3.private static int DEFAULT\_BUFFER\_SIZE = 8192; //默认容量

4.protected volatile InputStream in; //节点流

5.protected int marklimit; //

6.protected int markpos = -1; //标记的位置

7.private static int MAX\_BUFFER\_SIZE = Integer.MAX\_VALUE - 8; //最大容量

8.protected int pos; //缓冲中下一个要读的位置

构造方法：

1.public BufferedInputStream(InputStream in) //缓冲采用默认容量的byte数组

2.public BufferedInputStream(InputStream in, int size) //指定缓冲数组的容量

成员方法：

1. public synchronized int available() //还有多少字节未读
2. public void close() //方法内部关闭了节点流，因此只需要调用这个方法关闭包装流。
3. private void fill() //当缓冲中没有字节可读时会调用这个方法(pos >= count)。如果没有标记位置(markpos<0)则通过节点流读取数据到buf中，读满buf，覆盖buf原来的数据。待补充...
4. public synchronized void mark(int readlimit) //markpos记录当前pos，marklimit记录readlimit
5. public boolean markSupported() //支持reset和mark 默认返回true
6. public synchronized int read() //如果pos>=cout，说明缓冲读到头了，需要fill。否则直接从缓冲中返回pos位置的元素。
7. public synchronized void reset() //返回到标记的位置 pos=markpos
8. public synchronized long skip(long n)

BufferedOutPutStream:继承自FilterOutputStream

属性：  
1.protected byte buf[];

2.protected int count; //数组中有效的字节数

3.protected OutputStream out; //节点流

构造方法：

1.public BufferedOutputStream(OutputStream out)//默认8192

2.public BufferedOutputStream(OutputStream out, int size) //要想创建追加形式的缓冲流，则传入的节点流out指定为追加方式即可。

成员方法：

1.public synchronized void flush() //将缓冲中剩余的数据writer到文件中。

2.public synchronized void write(int b) //将这个字节添加到缓冲中，如果缓冲满了则将缓冲write到文件，否则不会write。

3.public synchronized void write(byte b[], int off, int len) //如果要输出的字节个数比缓冲还大，则直接输出(不用缓冲)，并且刷新缓冲。如果要输出的字节个数比缓冲剩余的空间多，则刷新缓冲，最后将数组b拷贝到缓冲。

FileReader：继承自InputStreamReader转换流

说明：InputStreamReader是包装流但FileReader不是，因为InputStreamReader的属性是私有的。

构造方法：

1.public FileReader(File file)

2.public FileReader(String fileName)

成员方法：

1. public String getEncoding() //得到当前的解码格式(字节->字符)
2. 不支持mark、reset
3. public int read() //读一个字符
4. public int read(char cbuf[], int offset, int length)
5. public long skip(long n)

InputStreamReader：

属性：

1.private final StreamDecoder sd; //也是个包装流 继承了Reader

构造方法：

1. public InputStreamReader(InputStream in) //参数为字节流
2. public InputStreamReader(InputStream in, String charsetName) //第二个参数是解码方式
3. 其他方法和FileReader一样，因为FileReader没有重写这些方法。

OutputStreamWriter和InputStreamReader基本一致，有flush方法并且有实现，字符流输出时要flush，不然缓存中的数据没有输出。底层是StreamEncoder的flush。

BufferedReader：

有个char数组缓冲，和字节缓冲流的实现原理类似。多了个readLine方法。