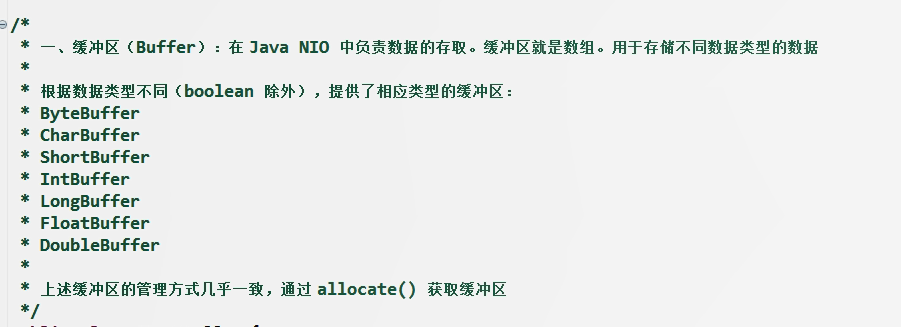
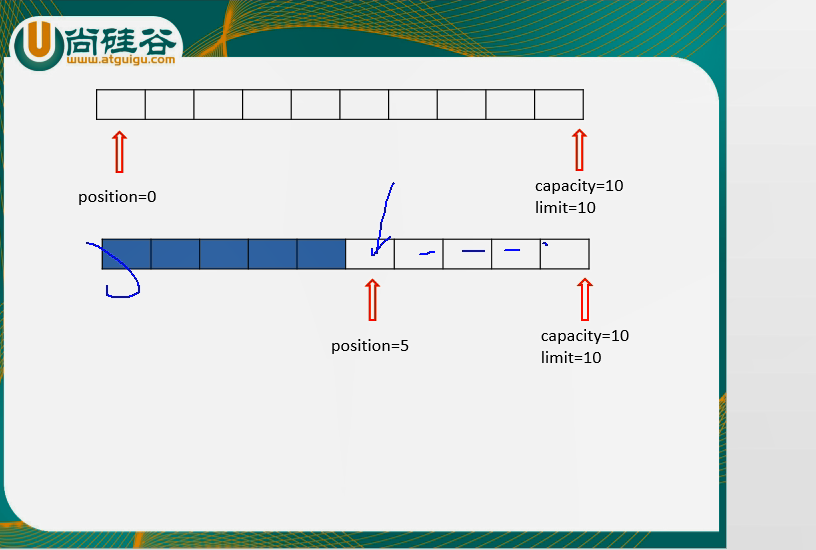
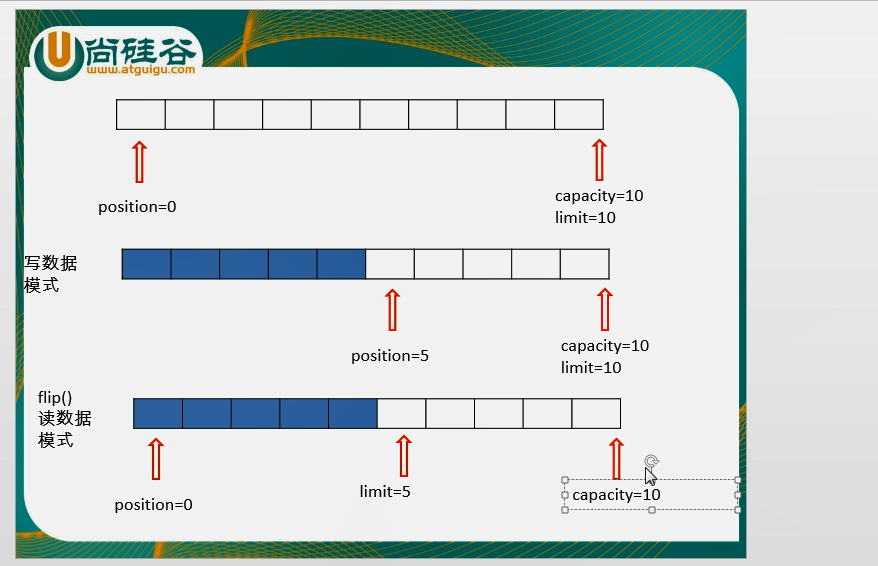
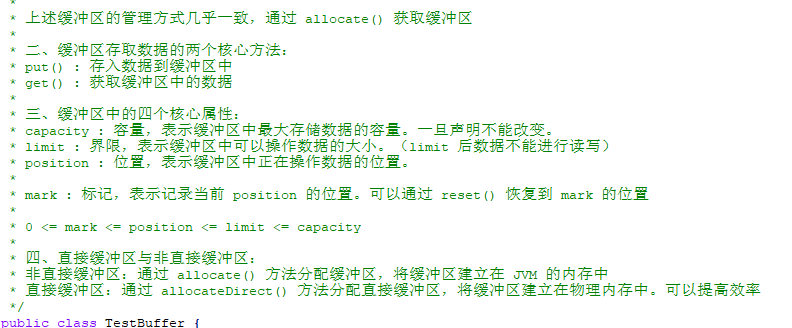
## Nio的缓冲区

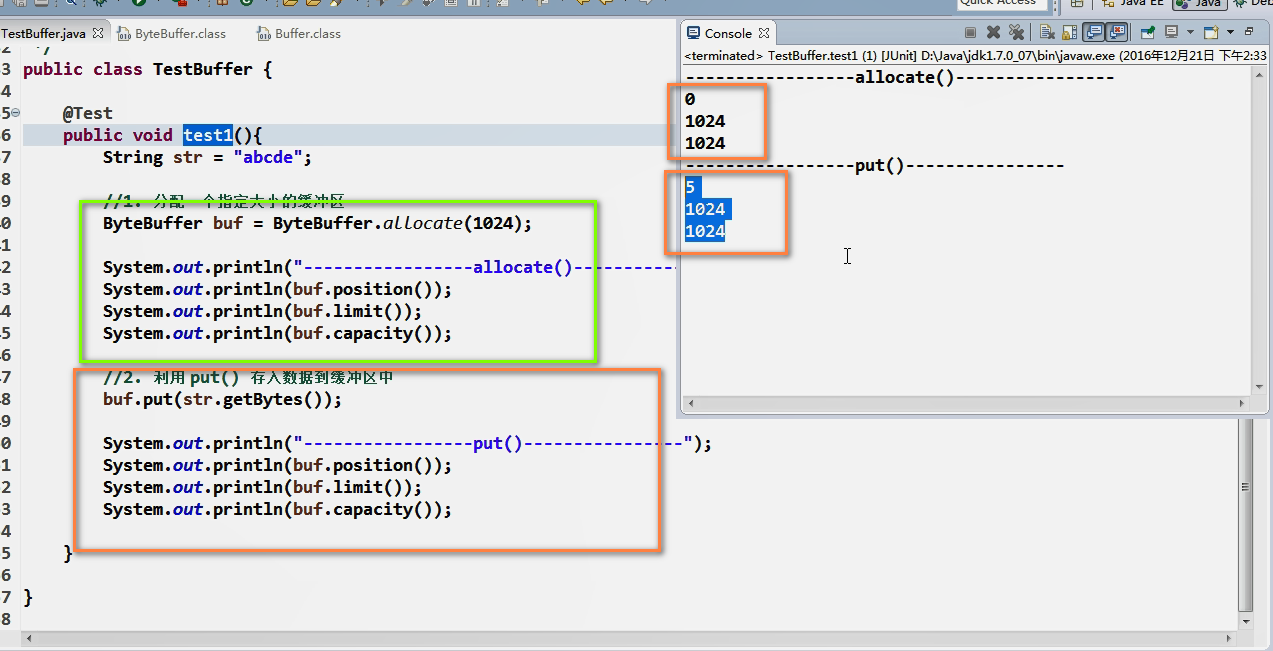


写入数据时候position位置按照数据长度变化

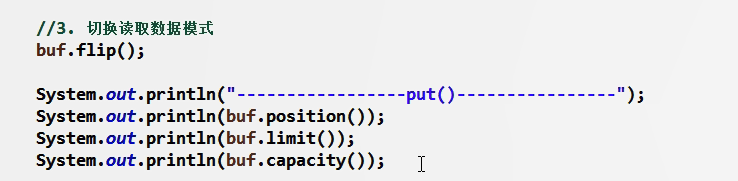


读数据的时候,调用flip()方法,position位置回到起始位置,limit到数据长度位置

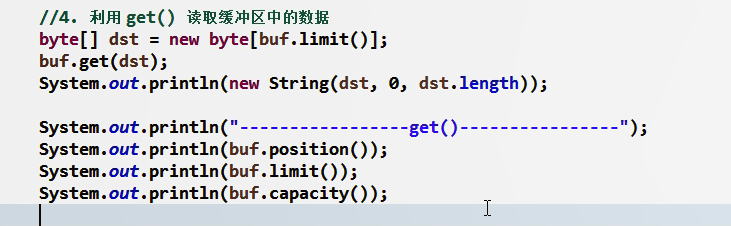




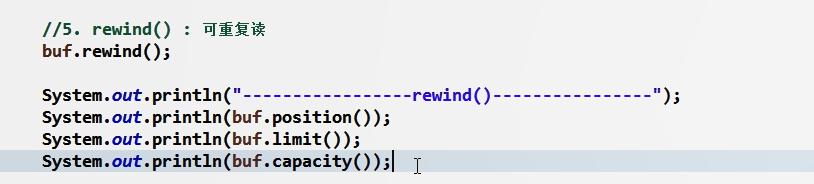
切换flip(),结果为0 5 1024



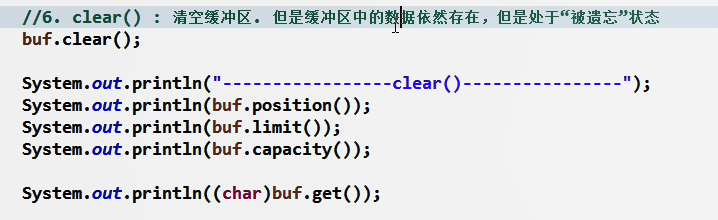
读取数据,结果为5 5 1024



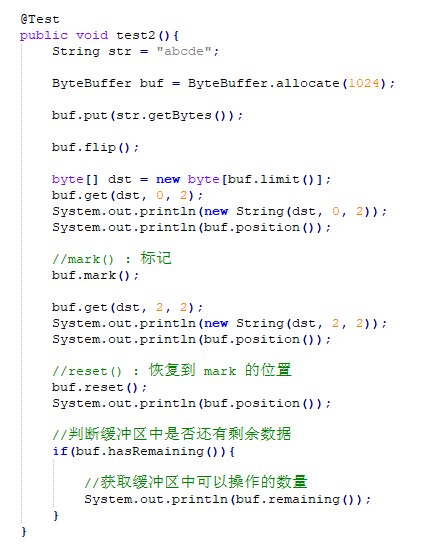
重复读数据,就是把positon位置复位 0 5 1024

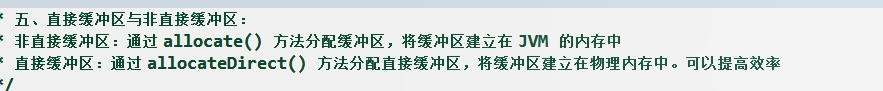


清空缓冲区,数据处于被遗忘状态,相当于复位limit的位置,实际数据还存在



mark : 标记，表示记录当前 position 的位置。可以通过 reset() 恢复到 mark 的位置





/\*

\* 一、通道（Channel）：用于源节点与目标节点的连接。在 Java NIO 中负责缓冲区中数据的传输。Channel 本身不存储数据，因此需要配合缓冲区进行传输。

\*

\* 二、通道的主要实现类

\* java.nio.channels.Channel 接口：

\* |--FileChannel

\* |--SocketChannel

\* |--ServerSocketChannel

\* |--DatagramChannel

\*

\* 三、获取通道

\* 1. Java 针对支持通道的类提供了 getChannel() 方法

\* 本地 IO：

\* FileInputStream/FileOutputStream

\* RandomAccessFile

\*

\* 网络IO：

\* Socket

\* ServerSocket

\* DatagramSocket

\*

\* 2. 在 JDK 1.7 中的 NIO.2 针对各个通道提供了静态方法 open()

\* 3. 在 JDK 1.7 中的 NIO.2 的 Files 工具类的 newByteChannel()

//利用通道完成文件的复制（非直接缓冲区）

@Test

public void test1(){//10874-10953

long start = System.currentTimeMillis();

FileInputStream fis = null;

FileOutputStream fos = null;

//①获取通道

FileChannel inChannel = null;

FileChannel outChannel = null;

try {

fis = new FileInputStream("d:/1.mkv");

fos = new FileOutputStream("d:/2.mkv");

//用输入输出流来获取管道.

inChannel = fis.getChannel();

outChannel = fos.getChannel();

//②分配指定大小的缓冲区

ByteBuffer buf = ByteBuffer.allocate(1024);

//③将通道中的数据存入缓冲区中

while(inChannel.read(buf) != -1){

buf.flip(); //切换读取数据的模式

//④将缓冲区中的数据写入通道中

outChannel.write(buf);

buf.clear(); //清空缓冲区

}

} catch (IOException e) {

e.printStackTrace();

} finally {

if(outChannel != null){

try {

outChannel.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if(inChannel != null){

try {

inChannel.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if(fos != null){

try {

fos.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if(fis != null){

try {

fis.close();

} catch (IOException e) {

e.printStackTrace();

}

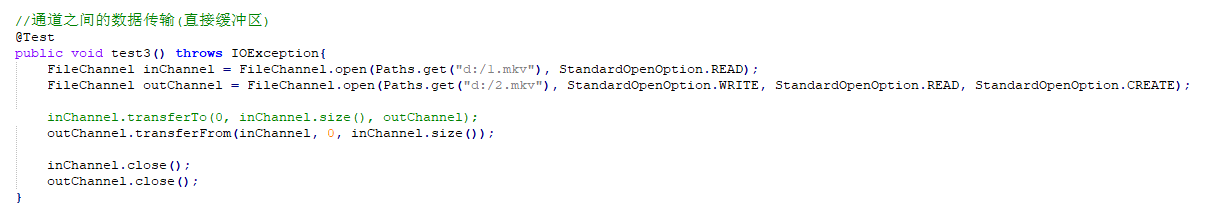
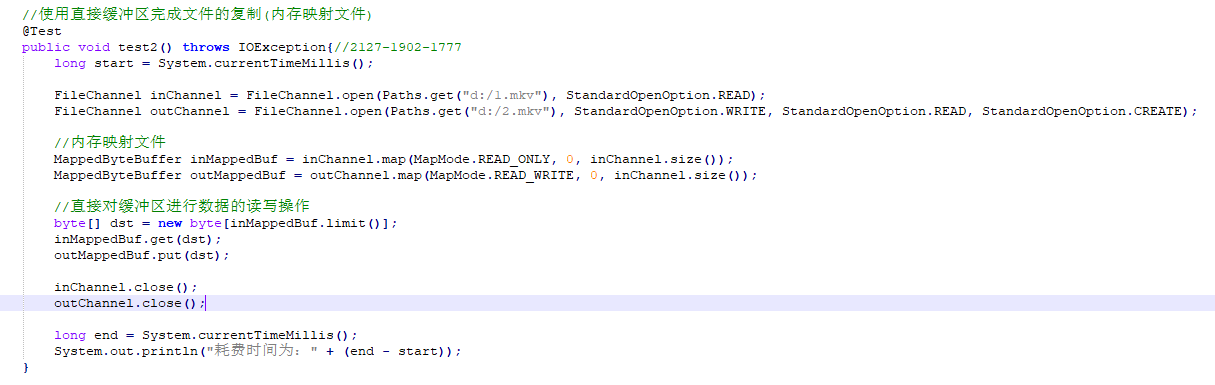
}

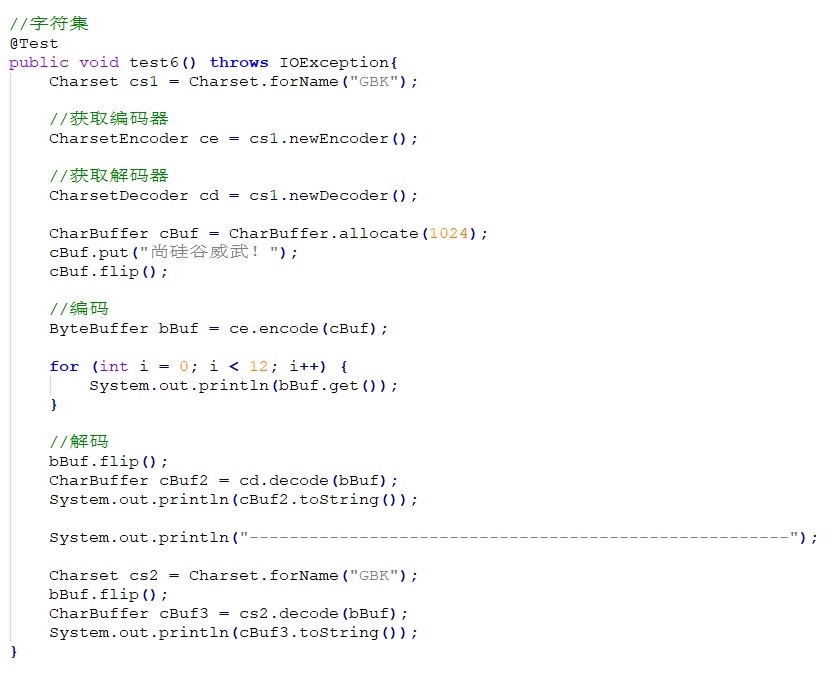
}

long end = System.currentTimeMillis();

System.out.println("耗费时间为：" + (end - start));

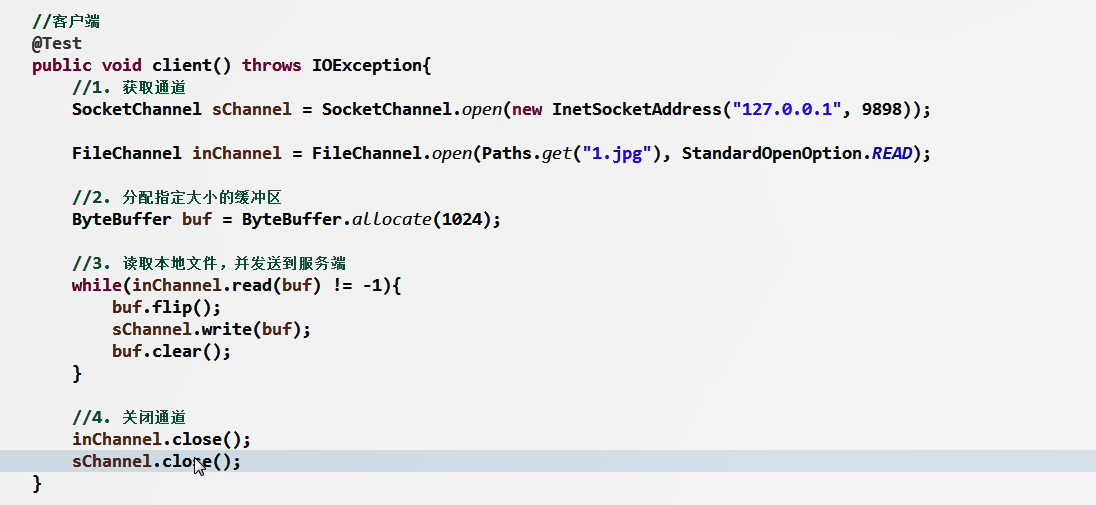
}

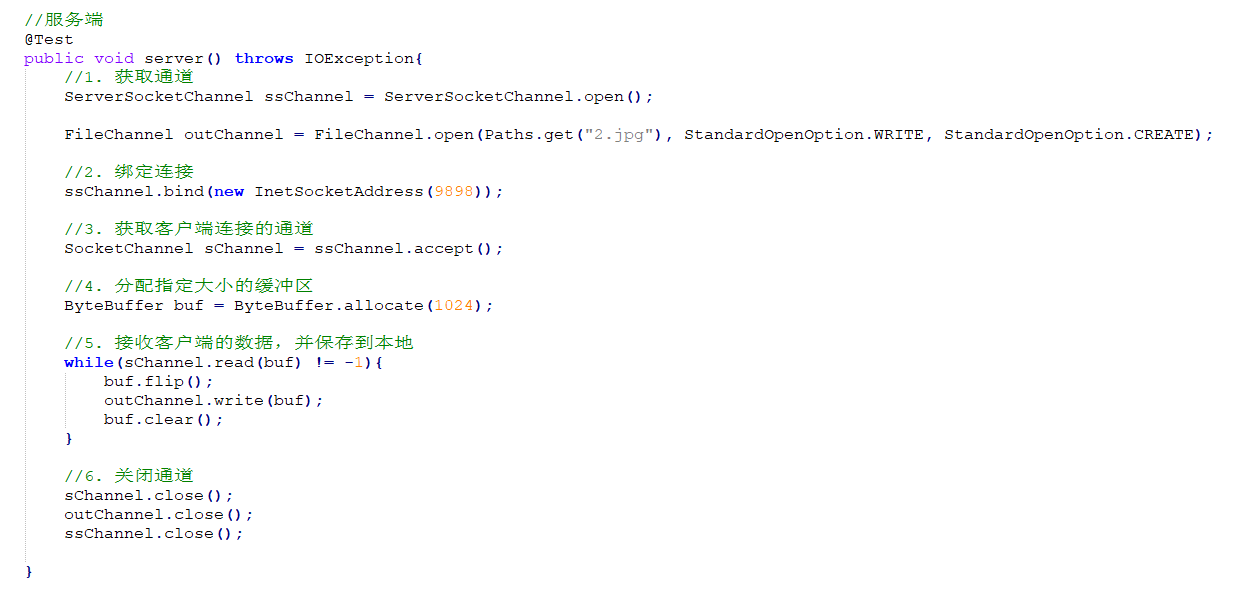




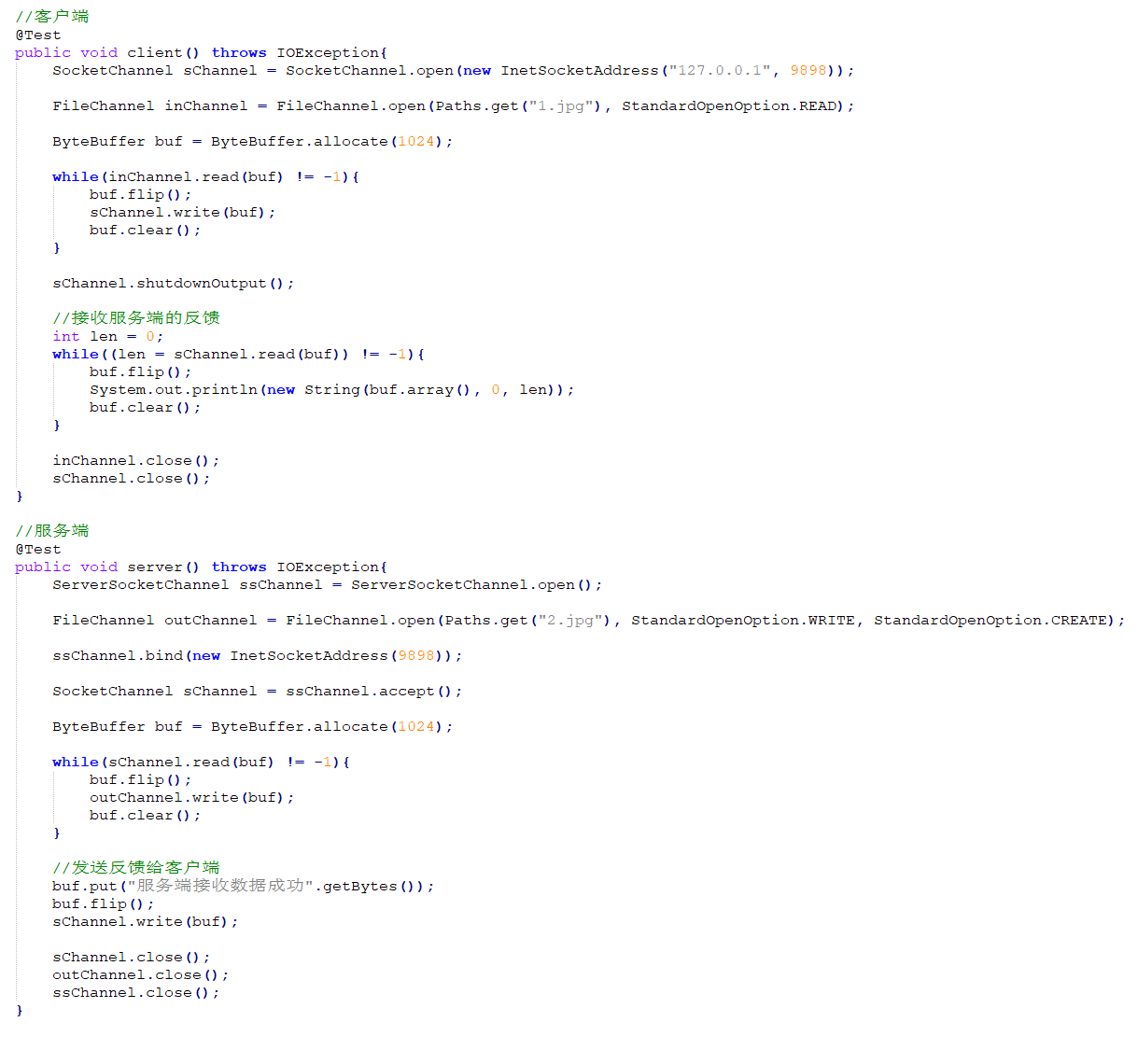
Filechannel不能设置成非阻塞模式

阻塞是io通信





客户端服务端相互通信,注意客户端的停止传输,否则会一直阻塞.



非阻塞式

