# java异步通信方式

实验名称：java网络通信编程基础

学时安排：2课时 指导老师：李赞

实验类别：验证型、设计型 实验要求：1人1组

学号： 姓名： 班级：

## 一、实验目的和任务

1. 本实验要求了解ubuntu平台java环境基本配置过程。

2. 能够在ubuntu平台运行基础的java程序。

3. 在ubuntu平台中运行简单的使用java编写的UDP通信程序。

## 二、实验设备介绍

1.软件需求： win10操作系统，VMware workstation，ubuntu18，java11。

## 三、注意事项和要求

1.设置Ubuntu软件源进行在线安装Java软件包。

2.java源程序不使用package，Java的文件名要与类名一样。

3.程序运行结果与源码截屏后提交到学习通平台。

## 四、实验内容和步骤

### **4.1基本的java程序**

java是面向对象的程序语言，每个java程序必须是一个类，所有的java类都继承自Object，每个java类的文件名必须与类名一样，文件后缀是.java，下面是一个最简单的java程序，该文件名是HelloWorld.java，类名也是HelloWorld。

public class HelloWorld {

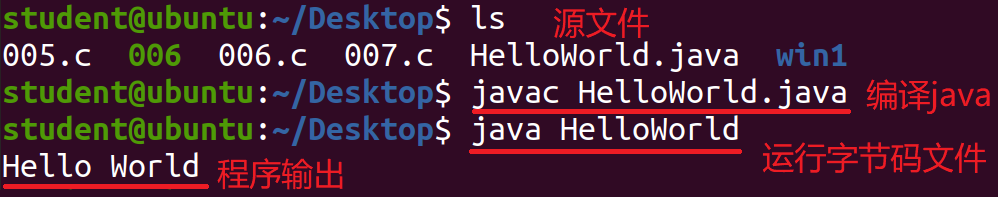
public static void main(String[] args) {

System.out.println("Hello World");

}

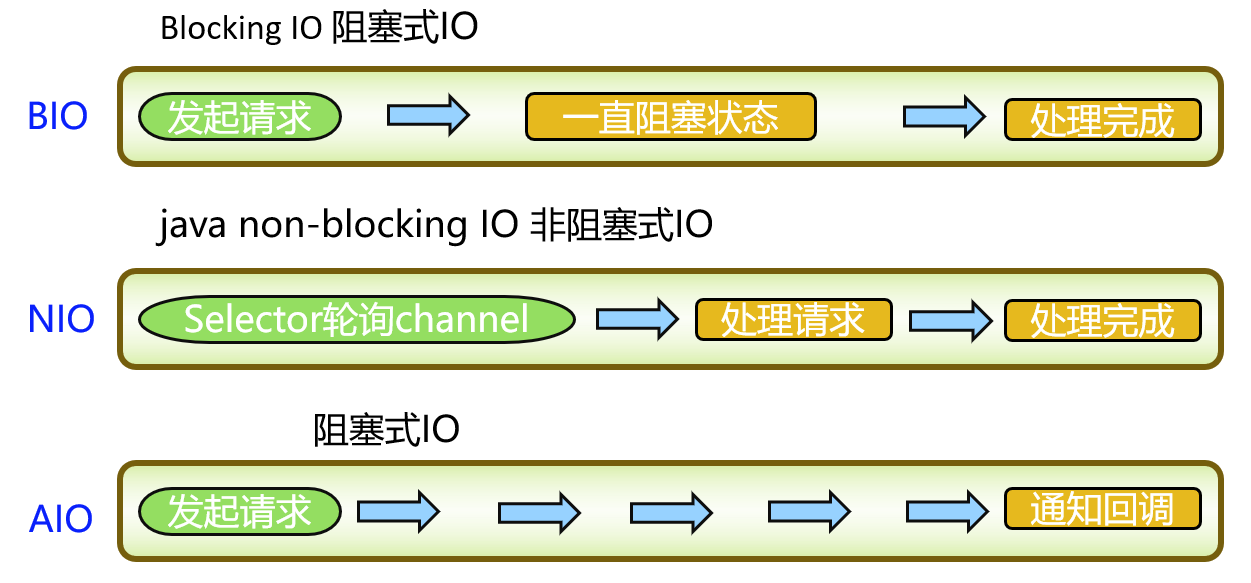
}

java程序需要使用javac命令对源码文件进行编译，并生成.class文件，该文件被称为字节码文件，并且要由java虚拟机来运行。



### **4.2JAVA的IO模式：BIO、NIO与AIO模式**

JAVA处理网通信中数据有三种模式：BIO、NIO和AIO，BIO指的是阻塞式IO，NIO是非阻塞式IO，AIO是异步非阻塞模型，三种模式对比如下图所示。



阻塞式IO是java1.4前使用的模式，程序调用的函数在其动作未完成会一直处于阻塞状态，后续的代码不会执行，例如从网络读取数据，代码会阻塞直至网络收到有可供读取的数据。阻塞式IO使用多线程模式实现多连接，程序运行效率不高。JAVA中基础的DatagramSocket， Socket，ServerSocket所使用的方法都是属于BIO。

I/O复用接口可以同时监听多个I/O事件以提高事件处理效率。Linux 环境中使用select/poll/epoll\_wait 实现I/O复用。java 1.4版本推出了一种新型的IO API代替标准java IO，实现相同的读写作用，称为NIO。NIO是面向缓冲区、基于通道的IO操作；通过NIO可以提高对文件的读写操作，具有较好优势，使用NIO的场景越来愈多，很多主流行的框架都使用到NIO技术，如Tomcat、Netty、Jetty等。

非阻塞式IO首先需要对IO事件进行注册，当特定 I/O 事件，如可读数据到达，新的套接字连接等等，在发生特定事件时，系统再通知程序。NIO 中实现非阻塞 I/O 的核心对象就是 Selector，Selector 就是注册各种 I/O 事件。NIO基于Reactor模式的工作方式，I/O 调用不会被阻塞，当有读或写等任何注册的事件发生时，从 Selector 中获得相应的SelectionKey，同时从 SelectionKey 中可以找到发生的事件和该事件所发生的具体的 SelectableChannel，以获得客户端发送过来的数据。

Java NIO 主要包含三个核心部分：Channels（通道）、Buffers（缓冲区）、Selectors（选择器），程序向Selector注册Channel，Selector运行单线程处理多个Channel，然后调用它的select()方法，事件包括：新的连接进来、数据接收等，当事件发生后select()方法返回。缓冲区与通道建立关联关系，channel连接缓冲区和IO设备（如：文件、套接字），程序读写缓冲区完成IO操作。NIO中主要包括下面四个Channel。

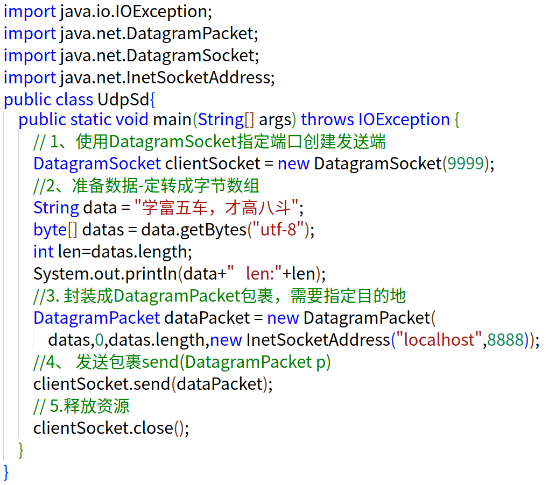
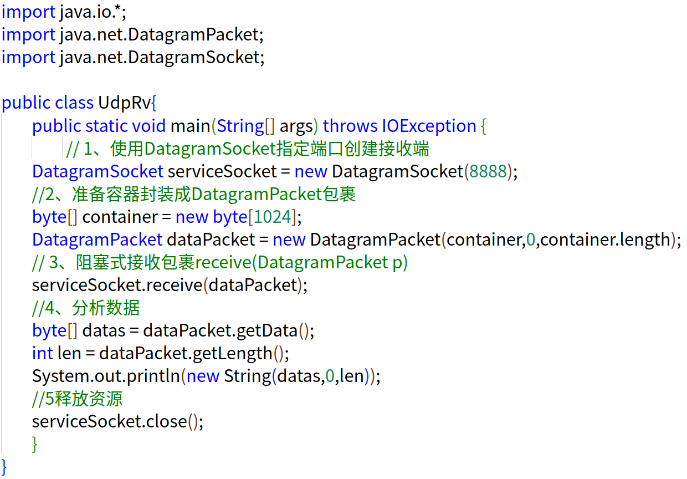
|  |  |  |
| --- | --- | --- |
| NIO通道名称 | 操作对象 | BIO对应类 |
| FileChannel | 文件 | File |
| DatagramChannel | UDP | DatagramSocket |
| SocketChannel | tcp客户端数据通信 | Socket |
| ServerSocketChannel | tcp服务端 | ServerSocket |

通道调用configureBlocking( )方法要设置成阻塞或非阻塞，参数值为true为阻塞模式，参数值为false为非阻塞模式，isBlocking( )方法来判断模式。

AIO模型是JDK1.7版本后中提供的异步非阻塞网络IO模型，AIO也被称作是NIO2.0。用户进程进行系统调用后可以处理其他逻辑，无论内核数据是否准备好，方法直接返回用户进程，进程不会因为系统调用阻塞。数据准备好，内核直接复制数据到用户空间，内核向进程发送通知。

### **4.3阻塞式的UDP程序**

下面是java程序实现的阻塞式的UDP通信。要传送的数据要封装到DatagramPacket对象中，发送和接收UDP数据使用DatagramSocket对象。DatagramSocket的send和receive方法都是阻塞式IO。



在两个命令窗口运行程序。

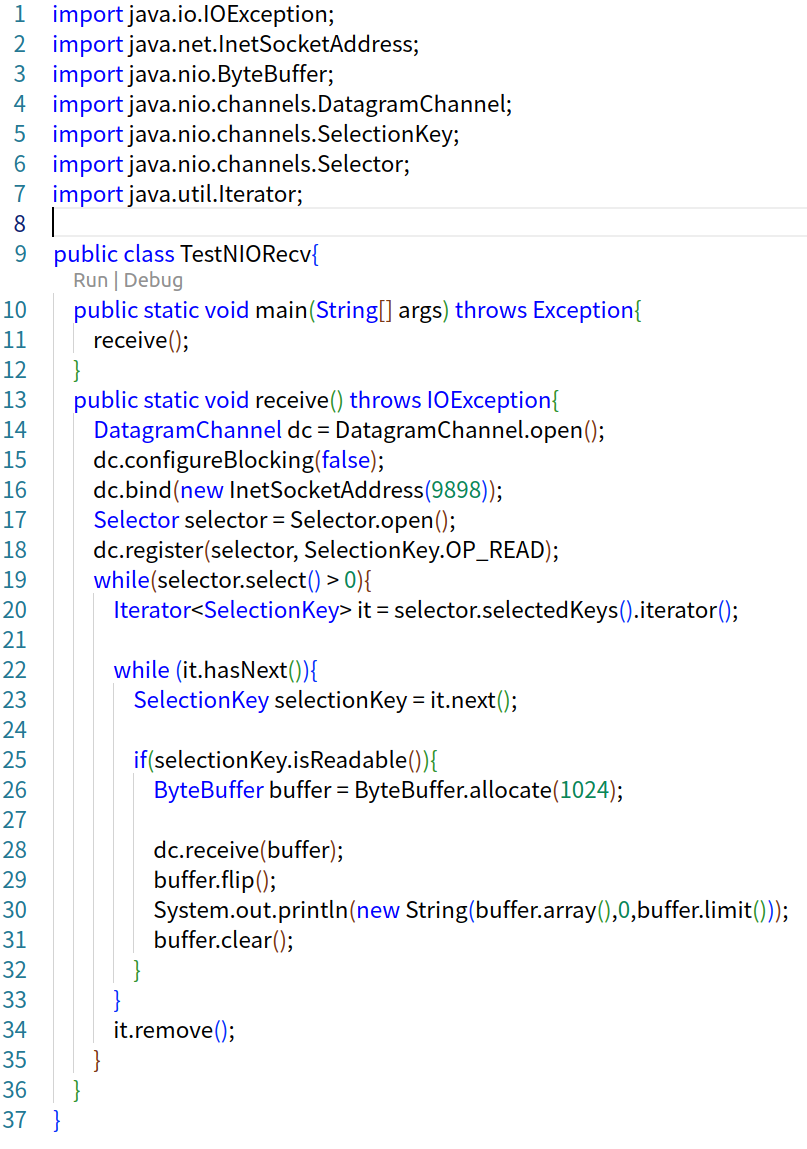


### **4.4NIO模式的UDP程序**

NIO模式主要通过相应的channel和缓冲区实现IO操作，UDP使用的是DatagramChannel对象。

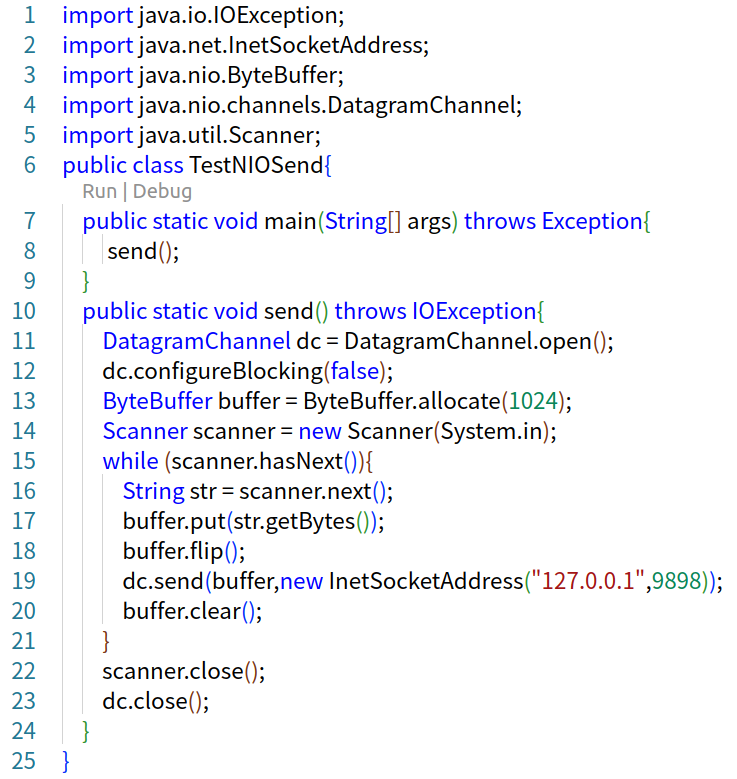
#### 4.4.1udp接收端

接收端创建DatagramChannel对象，经过bind端口绑定后，进行事件注册，通过缓冲区进行数据读取，将接收到的文本显示到控制台。



#### 4.4.2udp发送端

发送端使用于Scanner类与System.in关联后从输入流中扫描输入文本，然后把用户输入传入缓冲区然后通过Channel发送。



### **4.5NIO模式的TCP程序**

NIO使用SocketChannel对应的是Socket类，ServerSokcetChannel对应的是ServerSocket。

#### 4.5.1NIO tcp客户端

客户端用SocketChanel 对象实现TCP客户端，连接目标主机和端口从缓冲区写入和读取数据。

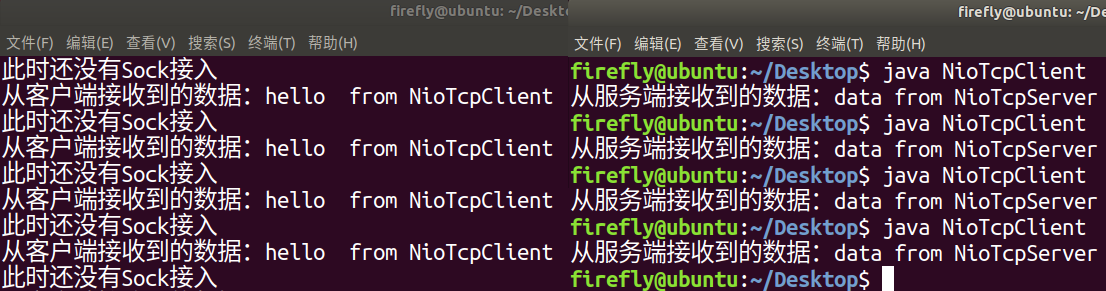


#### 4.5.2 NIO tcp服务端

服务端用ServerSocketChanel 对象实现TCP服务端，经过端口绑定，接收客户端连接，并通过SocketChannel对客户端进行数据读写。



下面是程序的运行结果演示。



### **4.6AIO模式的TCP程序**

AIO模式是在NIO基础上进一步将非阻塞式IO改造为异步工作模式，都使用ByteBuffer来缓存数据，Channel传输数据，读写方法是异步操作。

|  |  |  |
| --- | --- | --- |
| AIO异步通道类 | NIO通道类 | BIO对应类 |
| AsynchronousFileChannel | FileChannel | File |
| AsynchronousDatagramChannel | DatagramChannel | DatagramSocket |
| AsynchronousSocketChannel | SocketChannel | Socket |
| AsynchronousServerSocketChannel | ServerSocketChannel | ServerSocket |

#### 4.6.1AIO tcp客户端

这是实现简单Accept的AIO客户端的示例代码。



#### 4.6.2AIO tcp服务端

这是实现简单Accept的AIO服务端的示例代码。



下面是AIO的运行例子，



## 五、程序作业

1.编译和运行实验程序。

2. 使用实际IP地址调试并运行NIO模式的TCP程序，将截图提交到学习通平台。