**《面向对象程序设计(Java)》**

**实验报告**

姓 名 石家晖

学 号 2019081238

学 院 软件工程学院

专 业 软件工程

年 级 2019级

指导教师 文立玉

**成都信息工程大学软件工程学院**

**2020年9月**

# 实验9：

1. **实验目的**
   1. 理解Java网络模型。
   2. 掌握InetAddress类的使用。
   3. 理解Socket的相关概念、原理。
   4. 掌握Java Socket编程，学会创建Java网络服务器和网络客户机。
   5. 理解基于UDP协议的Socket的相关概念、原理。
   6. 掌握基于UDP协议的Socket编程的步骤及方法。
   7. 巩固Java GUI 编程和数据流的使用。
2. **实验要求**
3. 实验在2学时内完成。
4. 1人1组独立完成。
5. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1)设计思路：

基于 TCP 的 Socket 编程。使用套接字编写客户—服务器程序，实现客户 —服务器交互计算。客户将三角形 3 个边的长度发给服务器，服务器把计算 出的三角形的面积返回给客户。

(2)实验步骤

服务器端：Server.java

(a). 调用ServerSocket(int port)创建一个服务器端套接字，并绑定到指定端口上；

(b). 调用accept()，监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字；

(c). 调用Socket类的getOutputStream()和getInputStream获取输出流和输入流，使用输入流获取客户端传来的3条边的长度；

(d). 计算面积；

(e). 将面积通过输出流传递到客户端；

(f). 最后关闭通信套接字。

客户端Client.java

(a). 创建界面，在类的构造器中调用Socket()创建一个流套接字；

(b). 进行事件处理：按钮对象调用addActionListener注册监听，重写actionPerformed方法进行事件处理的具体操作

(c). 在事件处理过程中，客户端Socket连接到服务器端，调用Socket类的getOutputStream()和getInputStream获取输出流和输入流，开始网络数据的发送和接收。将三边传到服务器端，从服务器端获取计算出来的面积。最后关闭通信套接字。

(3) 关键代码

System.*out*.println("从客户端接收三角形的三边...");  
 for(int i = 0;i<a.length;i++){  
 line = in.readLine();  
 a[i] = Double.*parseDouble*(line);  
 }  
 double temp;  
 temp = testserver.Result(a);  
 System.*out*.println("三角形面积是"+ temp);

(4) 程序测试过程

①分别编译运行客户端和服务端

②在客户端输入测试样例

③检查服务端是否返回客户端正确结果

(5) 运行结果（运行效果）

2.

(1)设计思路：

基于UDP的Socket编程。编写一个基于UDP协议的Socket网络通信应用程序，实现如下功能：

1. 客户端能够发送任意的信息给服务器端。
2. 服务器端将收到的字符串变成大写后返还给客户端。

(2)实验步骤

接收端

(a). 调用DatagramSocket(int port)创建一个数据报套接字，并绑定到指定端口上；

(b). 调用DatagramPacket(byte[] buf, int length)，建立一个字节数组以接收UDP包；

(c). 调用DatagramSocket类的receive()，接收UDP包；

(d). 最后关闭数据报套接字。

发送端

(a). 调用DatagramSocket()创建一个数据报套接字；

(b). 调用DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)，建立要发送的UDP包；

(c). 调用DatagramSocket类的send()，发送UDP包；

(d). 最后关闭数据报套接字

(3) 关键代码

try{  
 System.*out*.println("服务已启动，正在等待接收数据包...");  
 DatagramSocket socket = new DatagramSocket(4470); //建立数据报套接字  
 DatagramPacket receivePacket = new DatagramPacket(revive,revive.length); //建立接收数据包  
 socket.receive(receivePacket); //从receivePacket接收数据  
 String str = new String(receivePacket.getData(),0,receivePacket.getLength());  
 System.*out*.println("客户端说:"+str);  
 InetAddress IP = receivePacket.getAddress(); //获取地址  
 int port = receivePacket.getPort();  
 String s = str.toUpperCase(); //转换为大写  
 send = s.getBytes();  
 DatagramPacket sendTo = new DatagramPacket(send,send.length,IP,port);  
 socket.send(sendTo);  
 } catch (SocketException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }

(4) 程序测试过程

①分别编译运行客户端和服务端

②在客户端输入测试样例

③检查服务端是否返回客户端正确结果

(5) 运行结果（运行效果）

1. **实验总结**

本次的实验主要涉及了JAVA网络编程相关的一些知识，其中基于TCP以及UDP的编程。在关于服务端与客服端之间的联系，客服端的输入，就对应于服务端的输出，反之类似，了解其大致原理之后，运用了BufferedReader和PrintWr来对其输入输出进行处理，方便了字符的读与写。DatagramSocket以及DatagramPacket的联合用法也可以让socket进行服务端与客服端的联系，当然用完方法之后，一定要记得close()。

1. **附录（程序代码）**

1.

Client:

package nine1;

import java.io.\*;

import java.net.\*;

public class Client {

public static void main(String[] args) {

try {

Socket socket = new Socket(InetAddress.getByName(null),10);//Socket

BufferedReader read = new BufferedReader(new InputStreamReader(System.in));//BufferedReader

PrintWriter PW = new PrintWriter(socket.getOutputStream());

System.out.println("从Server接收三角形的三条边发送到服务器:");

System.out.println("请输入3条边:");

for (int i = 0; i < 3; i++) {

PW.flush();

PW.println(read.readLine());

}

PW.close();

socket.close();

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Server:

package nine1;

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String args[]) {

Server tcp = new Server();

double a[] = new double[3];

try {

ServerSocket server = new ServerSocket(10);//ServerSocket

Socket client=server.accept();//Socket

BufferedReader BR = new BufferedReader(new InputStreamReader

(client.getInputStream()));//读取流

System.out.println("从Client接收三角形的三条边:");

String readline = "";

for (int i = 0; i < 3; i++) {

readline=BR.readLine();

a[i] = Double.valueOf(readline);

}

System.out.println("三角形的面积是:" + tcp.jiusan(a));

BR.close();//关闭读取流

client.close();//关闭Socket

server.close();//关闭ServerSocket

} catch (IOException e) {

e.printStackTrace();

};

}

public double jiusan(double a[]){//计算三角形面积的方法

double area = 0.0;

double temp = 0.0;

double aside = a[0];

double bside = a[1];

double cside = a[2];

temp = a[0] + a[1] + a[2];

temp = temp / 2;

area = Math.sqrt(temp \* (temp - aside) \* (temp - bside) \* (temp - cside));

return area;

}

}

2.

Server2:

package nine2;

import java.net.\*;

public class Server2{

public static void main(String[] args){

DatagramPacket bag;//DatagramPacket

DatagramSocket geter;//DatagramSocket

String str;

try {

geter = new DatagramSocket(2001);

byte[] array = new byte[1000];

System.out.println("服务器已启动,等待中waiting...");

bag = new DatagramPacket(array, array.length);

geter.receive(bag);

str = new String(bag.getData(),0,bag.getLength());

System.out.print("客户说:");

System.out.println(str);

byte[] arrout = new byte[1000];

arrout = str.toUpperCase().getBytes();

InetAddress addr = InetAddress.getByName("127.0.0.1");

DatagramPacket bagger = new DatagramPacket(arrout, arrout.length,addr,722);

geter.send(bagger);

geter.close();

}catch(Exception e) {System.out.println(e);}

}

}

Client2：

package nine2;

import java.util.Scanner;

import java.io.IOException;

import java.net.\*;

public class Client2 {

public static void main(String[] args) throws IOException{

Scanner in = new Scanner(System.in);

String string = null;

DatagramPacket DP;//DatagramPacket

DatagramSocket DS;//DatagramSocket

try {

byte[] arr = new byte[1000];//定义数据结构

System.out.print("请输入要说的话（字符串）:");

string = in.next();

arr = string.getBytes();

DP = new DatagramPacket(arr, arr.length,InetAddress.getByName(null),2001);//发送包

DS = new DatagramSocket();

DS.send(DP);

}catch(Exception e) {}

try {

byte[] bt = new byte[1000];

DatagramPacket dataPackIn = new DatagramPacket(bt, bt.length);

DS = new DatagramSocket(722);

DS.receive(dataPackIn);

string = new String(dataPackIn.getData(),0,dataPackIn.getLength());

System.out.print("服务器端返回的信息：" + string);

DS.close();

}catch(Exception e) {

System.out.println(e);

}

in.close();

}

}