网络程序设计基础实验报告

兰州大学信息科学与工程学院 2020级计算机科学与技术2班 徐宇奇 320190902531

第一部分

1.1 第一题

getOutputStream()方法用于获取流程和子流程的输出流。
getInputStream方法可以得到一个输入流，客户端的Socket对象上的getInputStream方法得到输入流其实就是从服务器端发回的数据。
ServerSocket的accept()方法从连接请求队列中取出一个客户的连接请求，然后创建与客户连接的Socket对象，并将它返回。如果队列中没有连接请求，accept()方法就会一直等待，直到接收到了连接请求才返回。

接下来，服务器从Socket对象中获得输入流和输出流，就能与客户交换数据。当服务器正在进行发送数据的操作时，如果客户端断开了连接，那么服务器端会抛出一个IOException的子类SocketException异常

1.4 第四题

编码不一样的时候需要使用编码转换，否则会出现乱码

第二部分

2.2 第二题

源代码如下

```
package lesson8;

import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class JsClient extends JFrame implements Runnable, ActionListener {
    JButton connection, jsbutton;
    JTextField inputA, inputB, inputC;
    JTextArea showResult;
    Socket socket;
    DataInputStream in = null;
    DataOutputStream out = null;
    Thread thread;

    public JsClient() {
        socket = new Socket();
        connection = new JButton("连接服务器");
        jsbutton = new JButton("求三角形面积");
```

```
        inputA = new JTextField("0", 12);
        inputB = new JTextField("0", 12);
        inputC = new JTextField("0", 12);
        Box boxV1 = Box.createVerticalBox();
        boxV1.add(new JLabel("输入边A"));
        boxV1.add(new JLabel("输入边B"));
        boxV1.add(new JLabel("输入边C"));
        Box boxV2 = Box.createVerticalBox();
        boxV2.add(inputA);
        boxV2.add(inputB);
        boxV2.add(inputC);
        Box baseBox = Box.createHorizontalBox();
        baseBox.add(boxV1);
        baseBox.add(boxV2);
        Container con = getContentPane();
        con.setLayout(new FlowLayout());
        showResult = new JTextArea(8, 18);
        con.add(connection);
        con.add(baseBox);
        con.add(jsbutton);
        con.add(new JScrollPane(showResult));
        jsbutton.addActionListener(this);
        connection.addActionListener(this);
        thread = new Thread(this);
        setBounds(100, 100, 360, 310);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void run() {
        while (true) {
            try {
                double area = in.readDouble();
                showResult.append("\n三角形的面积: \n" + area);
                showResult.setCaretPosition((showResult.getText()).length());
            } catch (IOException e) {
                showResult.setText("与服务器已断开");
                jsbutton.setEnabled(false);
                break;
            }
        }
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == connection) {
            try {
                if (socket.isConnected()) {
                } else {
                    InetAddress address = InetAddress.getByName("127.0.0.1");
                    InetSocketAddress socketAddress = new
InetSocketAddress(InetAddress.getLocalHost(), 4444);
                    socket.connect(socketAddress);
                    in = new DataInputStream(socket.getInputStream());
                    out = new DataOutputStream(socket.getOutputStream());
                    jsbutton.setEnabled(true);
```

```
                    thread.start();
                }
            } catch (IOException ee) {
            }
        }
        if (e.getSource() == jsbutton) {
            try {
                double a = Double.parseDouble(inputA.getText()),
                        b = Double.parseDouble(inputB.getText()),
                        c = Double.parseDouble(inputC.getText());
                if (a + b > c && a + c > b && b + c > a) {
                    out.writeDouble(a);
                    out.writeDouble(b);
                    out.writeDouble(c);
                } else {
                    inputA.setText("你输入的三个数不构成三角形");
                }
            } catch (Exception ee) {
                inputA.setText("请输入数字字符");
            }
        }
    }

    public static void main(String args[]) {
        JsClient win = new JsClient();
    }
}
```

```
package lesson8;

import java.io.*;
import java.net.*;

/**
 * @author LucasXu
 */
public class JsServer {
    public static void main(String args[]) {
        ServerSocket server = null;
        ServerThread thread;
        Socket client = null;
        while (true) {
            try {
                server = new ServerSocket(4444);
            } catch (IOException e1) {
                System.out.println("正在监听");
            }
            try {
                client = server.accept();
                System.out.println("客户的地址" + client.getInetAddress());
            } catch (IOException e) {
                System.out.println("正在等待客户");
            }
            if (client != null) {
                new ServerThread(client).start();
```

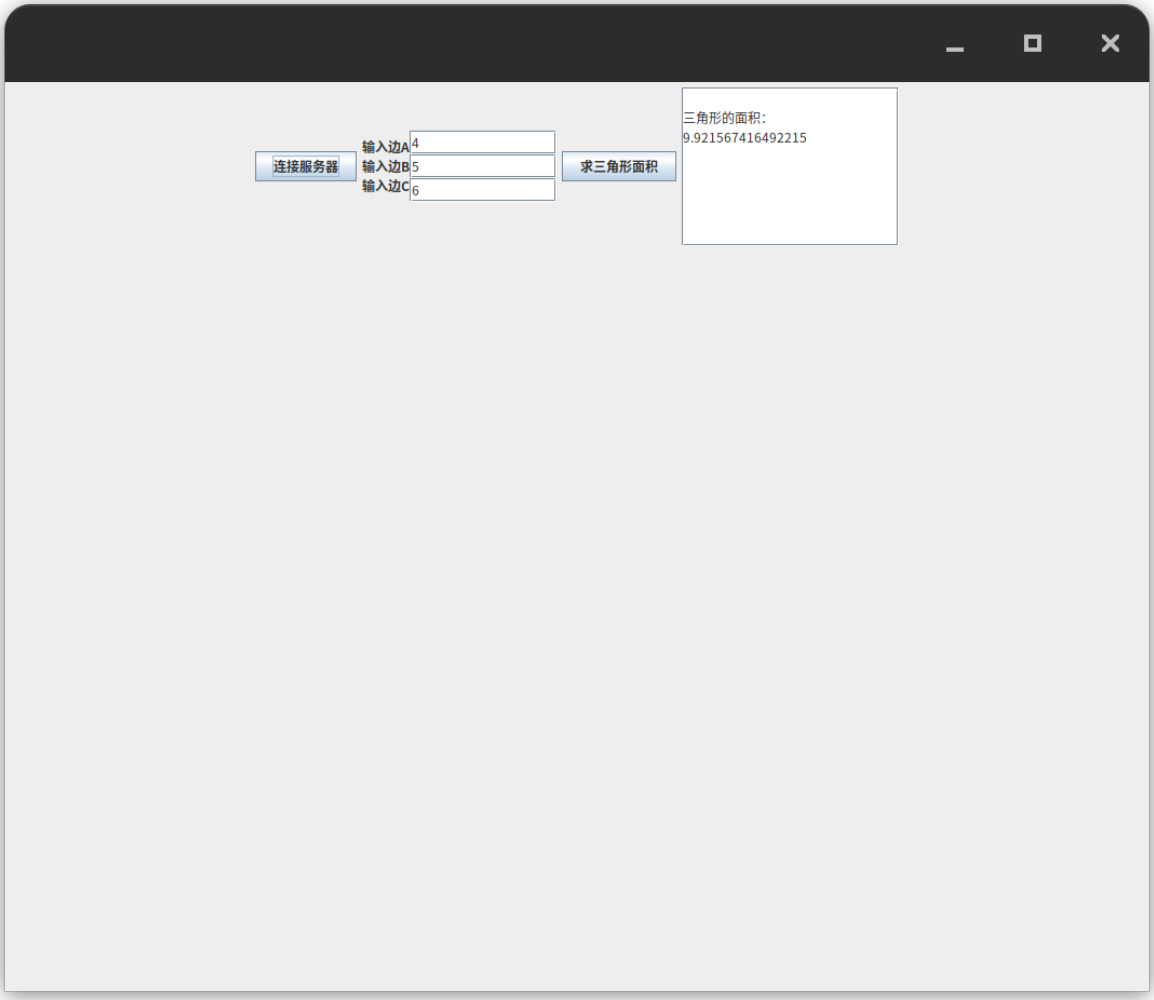
```
            } else {
                continue;
            }
        }
    }
}

class ServerThread extends Thread {
    Socket socket;
    DataOutputStream out = null;
    DataInputStream in = null;
    String s = null;

    ServerThread(Socket t) {
        socket = t;
        try {
            in = new DataInputStream(socket.getInputStream());
            out = new DataOutputStream(socket.getOutputStream());
        } catch (IOException e) {
        }
    }

    public void run() {
        while (true) {
            double a = 0, b = 0, c = 0, area = 0;
            try {
                a = in.readDouble();
                b = in.readDouble();
                c = in.readDouble();
                double p = (a + b + c) / 2.0;
                area = Math.sqrt(p * (p - a) * (p - b) * (p - c));
                out.writeDouble(area);
            } catch (IOException e) {
                System.out.println("客户离开");
                break;
            }
        }
    }
}
```

运行结果如下：



第三部分

聊天程序代码如下

```
package lesson8.ChatRoom;

import java.awt.Button;
import java.awt.Event;
import java.awt.Frame;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;

//客户端程序
public class ChatClient extends Frame {
    private static final long serialVersionUID = 1L;
    // 聊天室ID
    private String groupID;
    // 客户端用户名
```

```
    private String clientName;
    // 客户端消息发送服务套接字
    private DatagramSocket msg_send;
    // 服务端口
    private final int PORT = 10000;
    // 服务器IP地址
    private InetAddress ip;

    // 客户端控件
    TextField tf = new TextField(20);
    TextArea ta = new TextArea();
    Button send = new Button("send");

    // 客户端构造器
    public ChatClient(String groupID, String clientName) {

        super("聊天室:" + groupID + "/" + clientName);
        this.clientName = clientName;
        this.groupID = groupID;
        // 设置客户端界面样式
        add("North", tf);
        add("Center", ta);
        add("South", send);
        setSize(250, 250);
        show();
        // 聊天相关服务器初始化
        init();

        // 监视器
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                // 关闭消息发送服务
                msg_send.close();
                // 关闭客户端程序
                dispose();
                System.exit(0);
            }
        });

    }

    // 聊天相关服务器初始化
    private void init() {
        // 注册当前用户及所在聊天室信息注册到服务器
        ChatServer.logInGroup(groupID, this);
        try {
            // 初始化消息发送套接字对象
            msg_send = new DatagramSocket();
            // 指定消息服务器
            try {
                ip = InetAddress.getByName("127.0.0.1");
            } catch (UnknownHostException e) {
                System.out.println("未知的主机异常..");
            }
        } catch (SocketException e) {
```

```
            System.out.println("套接字连接异常..");
        }
    }

    // 消息发送按钮时间监听
    public boolean action(Event evt, Object arg) {
        if (evt.target.equals(send)) {
            try {
                // 获取输入内容
                String content = tf.getText();
                // 发送消息
                send_message(content);
                // 清空聊天框
                tf.setText(null);
            } catch (Exception ioe) {
                System.out.print(ioe.getMessage());
            }
        }
        return true;
    }

    // 消息发送
    private void send_message(String content) {
        // 消息格式化(json格式)
        String message = messageFormat(content);
        // 将消息封装成UDP数据包
        byte[] buf = message.getBytes();
        DatagramPacket packet = new DatagramPacket(buf, buf.length, ip, PORT);

        try {
            // 通过UDP协议发送消息
            msg_send.send(packet);
        } catch (IOException e) {
            System.out.println("IO异常..");
        }
    }

    // 消息格式化
    private String messageFormat(String content) {
        StringBuffer buffer = new StringBuffer();
        buffer.append("{\"groupId\":").append("\"").append(groupID).append(
                "\",");
        buffer.append("\"userName\":\"").append(clientName).append("\",");
        buffer.append("\"text\":\"").append(content).append("\"}");

        return buffer.toString();

    }

    // 从服务器获取当前聊天室最新消息(回调..)
    public void pushBackMessage(MessageEntity me) {
        ta.append(me.getUserName() + ":" + me.getText());
        ta.append("\n");

    }
```

```
}
```

```
package lesson8.ChatRoom;


import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.ArrayList;
import java.util.HashMap;

import com.google.gson.Gson;
/**
 * @author lucas
 */
public class ChatServer extends Thread {
    // 程序占用端口号
    private static final int PORT = 10000;
    // 消息接受套接字对象
    private static DatagramSocket server = null;
    // 字典对象(Key：聊天室ID，Value：该聊天室下的客户端用户集合);
    private static HashMap<String, ArrayList<ChatClient>> groups = new
HashMap<String, ArrayList<ChatClient>>();

    // 构造器
    public ChatServer() {
        try {
            // 消息接受套接字对象的构造初始化
            server = new DatagramSocket(PORT);
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }

    // 注册聊天室新登录用户
    public static void logInGroup(String groupID, ChatClient client) {
        // 通过聊天室ID，获取该聊天室的所有在线用户
        ArrayList<ChatClient> clients = groups.get(groupID);
        if (clients == null) {
            clients = new ArrayList<ChatClient>();
        }
        // 将此次进入聊天室的用户登记
        clients.add(client);
        // 更新聊天室信息
        groups.put(groupID, clients);
    }

    // 循环接收消息
    @Override
    public void run() {
        while (true) {
            receiveMessage();
        }
    }
```

```
    private void receiveMessage() {
        // UDP数据包
        byte[] buf = new byte[1024];
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        while (true) {
            try {
                // 接受数据包
                server.receive(packet);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            // 解析数据包，获取聊天信息
            String content = new String(packet.getData(), 0, packet.getLength());

            // 通过第三方包解析json数据
            Gson gson = new Gson();
            MessageEntity me = gson.fromJson(content, MessageEntity.class);

            // 解析消息内容，通过聊天室ID，获取该聊天室的所有在线用户
            ArrayList<ChatClient> clients = groups.get(me.getGroupId());

            // 将接收到的消息推送回该聊天室的各个用户
            for (ChatClient client : clients) {
                client.pushBackMessage(me);
            }
        }
    }
}
```

```
package lesson8.ChatRoom;
public class MessageEntity {
    private String groupId;
    private String userName;
    private String text;

    public String getGroupId() {
        return groupId;
    }

    public void setGroupId(String groupId) {
        this.groupId = groupId;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getText() {
```

```
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

}
```

```
package lesson8.ChatRoom;

public class Test {
    public static void main(String[] args) {
        ChatServer r = new ChatServer();
        r.start();

        ChatClient c1 = new ChatClient("001", "小红");
        ChatClient c2 = new ChatClient("001", "小绿");
        ChatClient c3 = new ChatClient("002", "小黑");
    }
}
```

运行截图如下：