`

【Experiment Name】 Multi-thread chatter software

【Purpose】

1.Learn java multi-threaded programming

2.learn java network programming

3.Learn Java Graphical User Interface Design

【Experimental principle】

1.Thread subclass and Runnable interface programming

2.Use of DatagramPacket and DatagramSocket classes

3.Use of the KeyListener interface

【Experimental content】

Experiment content: Design a complete chat software according to the basic example program of the chat software given.

Lab Requirements: (Week 8)

(1) Two people cooperate with each other to complete the design of the chat software;

(2) **Interface design:** refer to commercial software such as WeChat/QQ;

Basic requirements: On the basis of the picture on the right, add 2 text boxes for inputting the IP address and port number of the other party;

(3) In the chat record display area, add the names of both parties in the chat;

(4) **Add keyboard event processing, press the ENTER key on the keyboard to realize the information sending function**

Tip: You can increase the keyboard response event processing of the sending text box component, and send data in the keyboard event processing method . Methods as below:

1. Implement the keyboard response interface

class MyExtendsJFrame extends JFrame implements ActionListener ,

`

Runnable,KeyListener{

2. Rewrite the three events of the keyboard response: ke yPressed keyReleased keyTyped

For example: public void keyPressed(KeyEvent e) {

if(e.getKeyCode()==KeyEvent.VK_ENTER)

{Send data} //Judge whether it is enter key, if yes, send it

}

3. Add the association between the text box component and the keyboard response event

textSend.addKeyListener(this);

(5) Finally, use eclipse to package the program into a jar file

Lab Requirements: (Week 9)

(1) Add a receiving thread. run() receiving function, modified to receive thread.

(2) In the main function, based on an object of a Runnable implementation class, create two sub-threads, named: File and Text respectively; the functions are: receive files and receive text.

(3) In the run() thread body, first determine the name of the current thread, create two different receiving objects according to different thread names, open two different receiving ports, and define two loop bodies for receiving documents and text.

(4) Add 2 buttons to the chat interface, one is to send files and the other is to send text. Send to the same target IP address, different receiving port numbers, realize the sending of files and texts respectively.

//ChatterSoft.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Chat extends JFrame implements Runnable, ActionListener,
        ItemListener {
    private static final long serialVersionUID = 1L;
```

```
private static final String M_seg = "\r\t\n";
JTextArea areaContent = new JTextArea();
JTextField fieldSelfIP = new JTextField("");
JTextField fieldOtherIP = new JTextField("");
JTextField fieldSelfName = new JTextField("");
JTextField fieldOtherName = new JTextField("");
JPasswordField fieldSelfPassword = new JPasswordField("");
JPasswordField fieldOtherPassword = new JPasswordField("");
JTextField fieldSentence = new JTextField();
JButton buttonSend = new JButton("Sending...");
JButton buttonSendFile = new JButton("Sending files...");
Thread rec20100;
Thread rec20111;
Thread rec20122;
Thread sendTread;
JCheckBox checkEncry = new JCheckBox("Recv and send encryption");
SimpleDateFormat form = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
boolean isEncry = false;
private String tfilePath = null;
private String tfileName = null;

public Chat() {
    try {
        InetAddress address;
        address = InetAddress.getLocalHost();
        fieldSelfIP.setText(address.getHostAddress());
    } catch (Exception e) {
        return;
    }

    JPanel south = new JPanel();

    south.setLayout(new BorderLayout(5, 15));

    JPanel southOfSouth = new JPanel();

    JPanel centerOfSouth = new JPanel();

    centerOfSouth.setLayout(new GridLayout(3, 2, 15, 15));

    southOfSouth.setLayout(new GridLayout(1, 1));

    JPanel[] p = new JPanel[7];
    for (int i = 0; i < 7; i++) {
        p[i] = new JPanel();
```

```
        p[i].setLayout(new BorderLayout());
    }
    p[0].add(BorderLayout.WEST, new JLabel("Local IP:"));
    p[0].add(BorderLayout.CENTER, fieldSelfIP);
    p[1].add(BorderLayout.WEST, new JLabel("Receiver IP: "));
    p[1].add(BorderLayout.CENTER, fieldOtherIP);
    p[2].add(BorderLayout.WEST, new JLabel("Sender Name: "));
    p[2].add(BorderLayout.CENTER, fieldSelfName);
    p[3].add(BorderLayout.WEST, new JLabel("Receiver Name: "));
    p[3].add(BorderLayout.CENTER, fieldOtherName);
    p[4].add(checkEncry);
    p[5].add(buttonSendFile);
    p[6].add(BorderLayout.CENTER, fieldSentence);
    p[6].add(BorderLayout.EAST, buttonSend);
    for (int i = 0; i < 6; i++) {
        centerOfSouth.add(p[i]);
    }

    southOfSouth.add(p[6]);

    south.add(centerOfSouth, BorderLayout.CENTER);

    south.add(southOfSouth, BorderLayout.SOUTH);

    Container con = this.getContentPane();

    con.add(south, BorderLayout.SOUTH);

    con.add(new JScrollPane(areaContent));

    areaContent.setEditable(false);

    fieldSelfIP.setEditable(false);

    buttonSend.addActionListener(this);

    buttonSendFile.addActionListener(this);

    checkEncry.addItemListener(this);
    checkEncry.setSelected(false);

    buttonSend.registerKeyboardAction(this,
            KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0, true),
            JComponent.WHEN_IN_FOCUSED_WINDOW);
    this.setTitle("ChatterSoft - UDP Port: 20100, 20122, 20133, TCP Port:
```

```
20111");

        final Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
        final int width = 550;
        final int height = 650;
        final int left = (screen.width - width) / 2;
        final int top = (screen.height - height) / 2;
        this.setLocation(left, top);
        this.setSize(width, height);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        rec20100 = new Thread(this);
        rec20100.setName("20100");
        rec20111 = new Thread(this);
        rec20111.setName("20111");
        rec20122 = new Thread(this);
        rec20122.setName("20122");
        sendTread = new Thread(this);
        sendTread.setName("send");
        setVisible(true);
    }

    public static byte[] setAndGetEncryption(byte[] buf) {
        int len = buf.length;
        for (int i = 0; i < len; i++) {
            buf[i] = (byte) (255 - buf[i]);
        }
        return buf;
    }

    public static void main(String[] args) {
        Chat c = new Chat();
        c.rec20100.start();
        c.rec20111.start();
        c.rec20122.start();
        c.sendTread.start();
    }

    public Socket connectSocketServer(String IP, int port) {
        try {
            Socket s = new Socket(InetAddress.getByName(IP), port);
            return s;
        } catch (UnknownHostException e) {
            e.printStackTrace();
            return null;
```

```
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    public void request() {
        DatagramSocket socketSend;
        try {
            socketSend = new DatagramSocket();
            byte[] buf = this.tfileName.getBytes();
            InetAddress otherAddress = InetAddress.getByName(this.fieldOtherIP
                    .getText());
            DatagramPacket packet = new DatagramPacket(buf, buf.length,
                    otherAddress, 20122);
            socketSend.send(packet);
        } catch (SocketException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public boolean tryToSend() {
        while (true) {
            try {
                DatagramSocket socketRecieve = new DatagramSocket(20133);
                while (true) {
                    byte[] buf = new byte[1024];
                    DatagramPacket packet = new DatagramPacket(buf,
buf.length);
                    socketRecieve.receive(packet);
                    InetAddress address = packet.getAddress();
                    String srcIP = address.getHostAddress();
                    if (!srcIP.equals(this.fieldOtherIP.getText())) {
                        continue;
                    }
                    int length = packet.getLength();
                    String message = new String(buf, 0, length);

                    String[] word = message.split(M_seg);
                    if (word == null || word.length != 2) {
                        continue;
                    }
                    if (!word[1].trim().equals(this.tfileName)) {
```

```
                    continue;
                }
                if (word[0].equals("accept")) {
                    if (sendFile()) {
                        JOptionPane.showMessageDialog(this, "File transfer
successfully!");
                        return true;
                    } else {
                        JOptionPane.showMessageDialog(this, "File transfer
failed!");
                        return false;
                    }
                } else {
                    JOptionPane.showMessageDialog(this, "Receiver reject the
file.");
                    return false;
                }
            }
        } catch (SocketException e) {
            e.printStackTrace();
            return false;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
}

public boolean sendFile() {
    Socket s = connectSocketServer(this.fieldOtherIP.getText(), 20111);
    byte[] b = new byte[1024];
    File f = new File(this.tfilePath);
    try {

        DataOutputStream dout = new DataOutputStream(
                new BufferedOutputStream(s.getOutputStream()));

        FileInputStream fr = new FileInputStream(f);
        int n = fr.read(b);
        while (n != -1) {

            dout.write(b, 0, n);
            dout.flush();

            n = fr.read(b);
```

```
            }

            fr.close();
            dout.close();
            return true;
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            return false;
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }

    public void listen20100() {
        try {
            DatagramSocket socketRecieve = new DatagramSocket(20100);
            while (true) {
                byte[] buf = new byte[1024];
                DatagramPacket packet = new DatagramPacket(buf, buf.length);
                socketRecieve.receive(packet);
                InetAddress address;
                address = packet.getAddress();
                String currentOtherIP = address.getHostAddress();
                int length = packet.getLength();

                if (isEncry) {
                    buf = Chat.setAndGetEncryption(buf);
                }

                String message = new String(buf, 0, length);
                String strOtherIP = fieldOtherIP.getText().trim();
                if (currentOtherIP.trim().equals("")) {
                    currentOtherIP = "???";
                }
                if (!strOtherIP.equals("") && !currentOtherIP.equals(strOtherIP)) {
                    continue;
                }

                String strOtherName = fieldOtherName.getText().trim();
                if (strOtherName.length() == 0) {
                    strOtherName = "???";
                }
                String strSentence = message;
```

```
                this.printlnRecMessage(currentOtherIP, strOtherName, strSentence);
            }
        } catch (SocketException e) {
        } catch (Exception e) {
        }
    }

    public boolean listen20111() {
        try {
            byte[] b = new byte[1024];
            ServerSocket ss = new ServerSocket(20111);
            while (true) {
                Socket s = ss.accept();

                InputStream in = s.getInputStream();
                DataInputStream din = new DataInputStream(
                        new BufferedInputStream(in));

                String filePath = "";
                try {
                    JFileChooser fileChooser = new JFileChooser(".");
                    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                    int n = fileChooser.showSaveDialog(this);
                    if (n == JFileChooser.APPROVE_OPTION) {
                        filePath = fileChooser.getSelectedFile().getPath();
                    }
                } catch (Exception ex) {
                    ex.printStackTrace();
                    return false;
                }

                File f = new File(filePath);
                RandomAccessFile fw = new RandomAccessFile(f, "rw");

                int num = din.read(b);
                while (num != -1) {
                    fw.write(b, 0, num);
                    fw.skipBytes(num);
                    num = din.read(b);
                }
                din.close();
                fw.close();
                s.close();
                JOptionPane.showMessageDialog(this,          "Receiving          file
successfully");
```

```
            }
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }

    public void response(boolean accept, String fileName, String aimIP) {
        DatagramSocket socketSend;
        try {
            socketSend = new DatagramSocket();
            String message = "";
            if (accept) {
                message += "accept" + M_seg + fileName;
            } else {
                message += "refuse" + M_seg + fileName;
            }

            byte[] buf = message.getBytes();
            InetAddress otherAddress = InetAddress.getByName(aimIP);
            DatagramPacket packet = new DatagramPacket(buf, buf.length,
                    otherAddress, 20133);
            socketSend.send(packet);
        } catch (SocketException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void listen20122() {
        String srcIP = null;
        try {
            DatagramSocket socketRecieve = new DatagramSocket(20122);
            while (true) {
                byte[] buf = new byte[1024];
                DatagramPacket packet = new DatagramPacket(buf, buf.length);
                socketRecieve.receive(packet);
                InetAddress address = packet.getAddress();
                srcIP = address.getHostAddress();

                int length = packet.getLength();
                String message = new String(buf, 0, length);

                int result = JOptionPane.showConfirmDialog(this, "Host " + srcIP
```

`

```
                        + " is asking your receiving privilege " + message + ",
accept the file or not?");
                if (result == JOptionPane.YES_OPTION) {
                    response(true, message, srcIP);
                } else {
                    response(false, message, srcIP);
                }
            }
        } catch (SocketException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void run() {
        String thread = Thread.currentThread().getName();
        if (thread.equals("20100")) {
            listen20100();
        } else if (thread.equals("20122")) {
            listen20122();
        } else if (thread.equals("20111")) {
            listen20111();
        } else if (thread.equals("send")) {
            tryToSend();
        }
    }

    public void actionPerformed(ActionEvent ae) {
        Object obj = ae.getSource();

        if (obj == this.buttonSendFile) {
            JFileChooser chooser = new JFileChooser(".");
            int returnVal = chooser.showOpenDialog(this);
            if (returnVal == JFileChooser.APPROVE_OPTION) {
                this.tfilePath = chooser.getSelectedFile().getAbsolutePath();
                this.tfileName = (new File(this.tfilePath)).getName();
                request();
            } else {
            }
            return;
        }

        if (obj == buttonSend) {
            try {
```

```
                DatagramSocket socketSend = new DatagramSocket();
                String strOtherIP = fieldOtherIP.getText().trim();
                if (strOtherIP.length() == 0) {
                    String temp = this.fieldSelfIP.getText();
                    int index = temp.lastIndexOf(".");
                    strOtherIP = temp.substring(0, index);
                    strOtherIP += ".255";
                    JOptionPane.showMessageDialog(this, "The receiver's ip is
empty. Using broadcasting method to send messages." + strOtherIP);
                }

                String strOtherName = fieldOtherName.getText().trim();
                if (strOtherName.length() == 0) {
                    strOtherName = "???";
                }

                String strSentence = fieldSentence.getText();
                if (strSentence.length() == 0) {
                    JOptionPane.showMessageDialog(this, "Posting content should
not be empty");
                    return;
                }

                this.printlnSendMessage(strOtherIP, strOtherName, strSentence);

                byte[] buf = strSentence.getBytes();

                if (isEncry) {
                    buf = Chat.setAndGetEncryption(buf);
                }

                InetAddress otherAddress;

                otherAddress = InetAddress.getByName(strOtherIP);

                DatagramPacket packet = new DatagramPacket(buf, buf.length,
                        otherAddress, 20100);
                socketSend.send(packet);
                fieldSentence.setText("");
            } catch (IOException e) {
            }
            return;
        }
```

`

```
    }

    private void printlnSendMessage(String strOtherIP, String strOtherName, String
strSentence) {

        String nowtimes = form.format(new Date());
        String show = nowtimes;
        show += "LocalHost" + "\n";

        show += strSentence + "\n";
        areaContent.append(show);
        areaContent.setCaretPosition(areaContent.getText().length());
    }

    private void printlnRecMessage(String srcIP, String strOtherName, String
strSentence) {
        String nowtimes = form.format(new Date());
        String show = nowtimes;
        show += strOtherName + "\n";
        show += strSentence;
        show += "\n";
        areaContent.append(show);
        areaContent.setCaretPosition(areaContent.getText().length());
    }

    @Override
    public void itemStateChanged(ItemEvent ie) {
        this.isEncry = checkEncry.isSelected();
    }
}
```