学号 X41714008, P71714026 专业 计算机科学与技术，计算机科学与技术（英才班） 姓名 李逢时，韩宇晨

实验日期 **19/4/21**  教师签字 成绩

实验报告

【实验名称】 实验七 多线程聊天软件

【实验目的】

1. 学习java多线程编程
2. 学习java网路编程
3. 学习java图形用户界面设计

【实验原理】

1. Thread子类和Runnable接口编程
2. DatagramPacket 和DatagramSocket类的使用
3. KeyListener接口的使用

【实验内容】

**实验内容:** 根据已给出的聊天软件的基础例子程序，设计一个完整的聊天软件。

**实验要求：（第8周）**

（1）两个人互相合作完成聊天软件的设计;

（2）**界面设计：**参考微信/QQ等商业软件；

基本要求：在右图的基础上,添加2个文本框,用于输入对方的IP地址和端口号;

（3）在聊天记录显示区,给聊天双方添加姓名；

（4）**添加键盘事件处理，按下键盘的ENTER键， 实现信息发送功能**

提示：可以增加发送文本框组件的键盘响应事件处理，在键盘事件处理方法中，发送数据。方法如下：

1. 实现键盘响应接口

class MyExtendsJFrame extends JFrame implements ActionListener , Runnable,KeyListener{

1. 重写键盘响应的三个事件keyPressed keyReleased keyTyped

例如: public void keyPressed(KeyEvent e) {

if( e.getKeyCode()==KeyEvent.VK\_ENTER)

{发送数据} //判断是否是enter健,如果是则发送

}

1. 添加文本框组件和键盘响应事件的关联

textSend.addKeyListener(this);

（5） 最后，使用eclipse把程序打包成jar文件

**实验要求：（第9周）**

（1）添加接收线程。run()接收函数，修改为接收线程。

（2）在main函数中，基于1个Runnable实现类的对象，创建2个子线程，分别命名为：File和Text；功能分别为：接收文件和接收文本。

（3）在run()线程体中，先判断当前线程的名字，根据不同的线程名，创建两个不同的接收对象，打开两个不同的接收端口，定义两个循环体，分别用于接收文件和文本。

（4）聊天界面中添加2个按钮，一个是发送文件，一个是发送文本。发送到同一个目标IP地址的，不同的接收端口号，分别实现文件和文本的发送。

**//Chatter.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("Chatter - 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

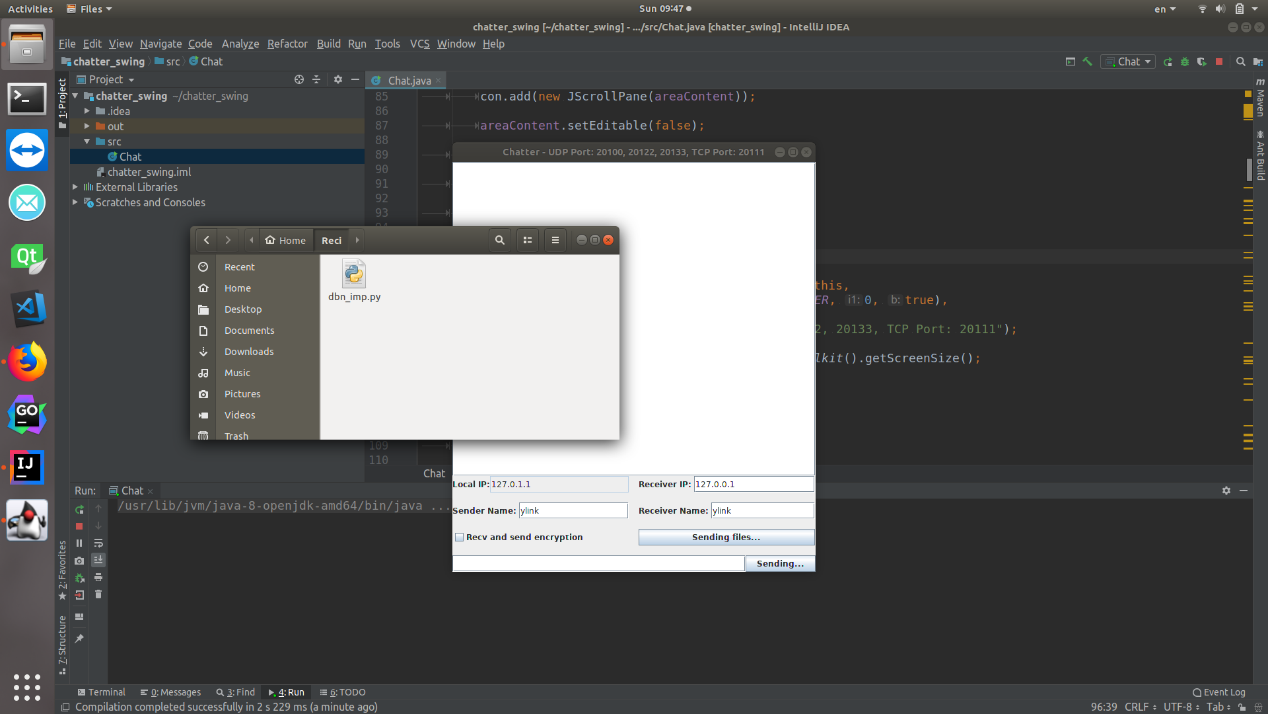
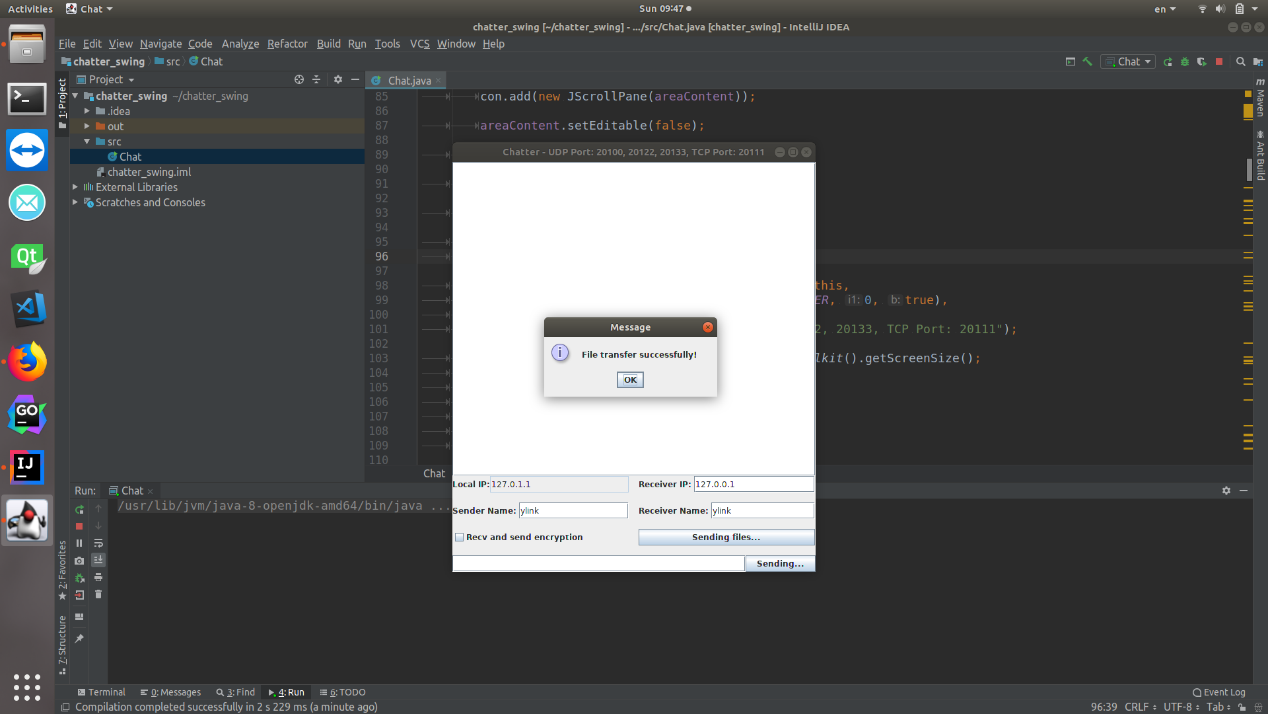
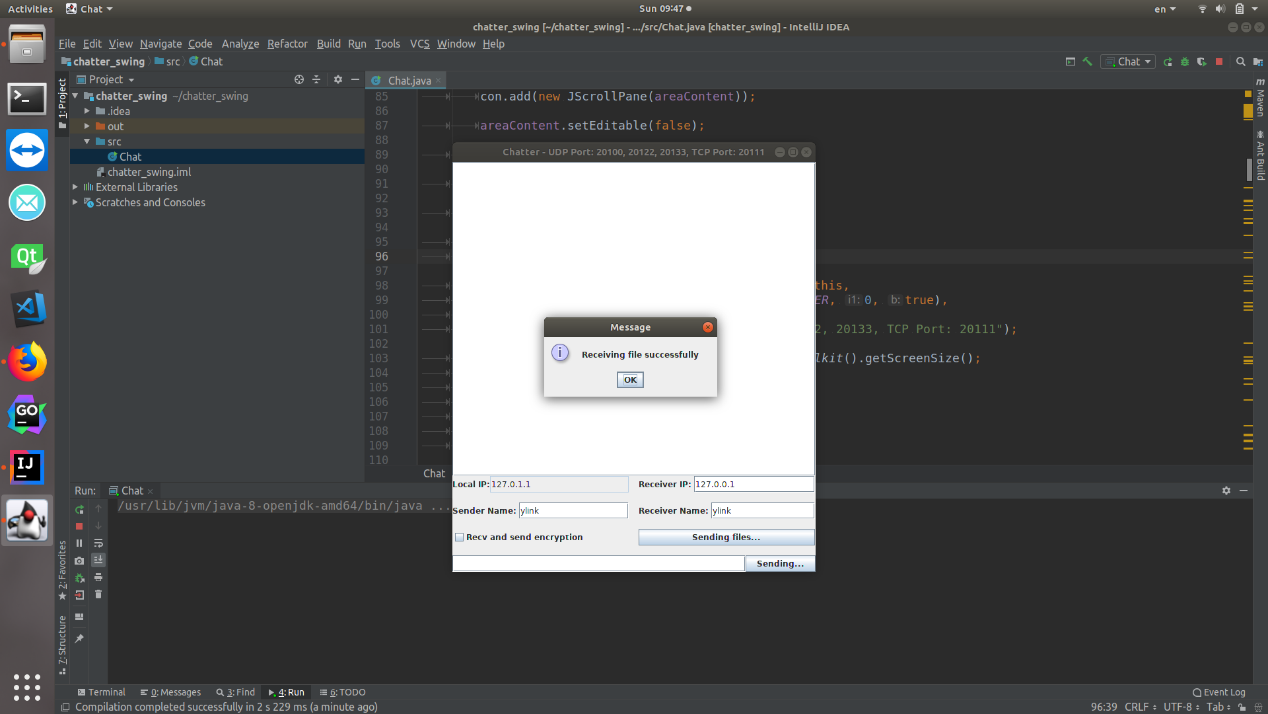
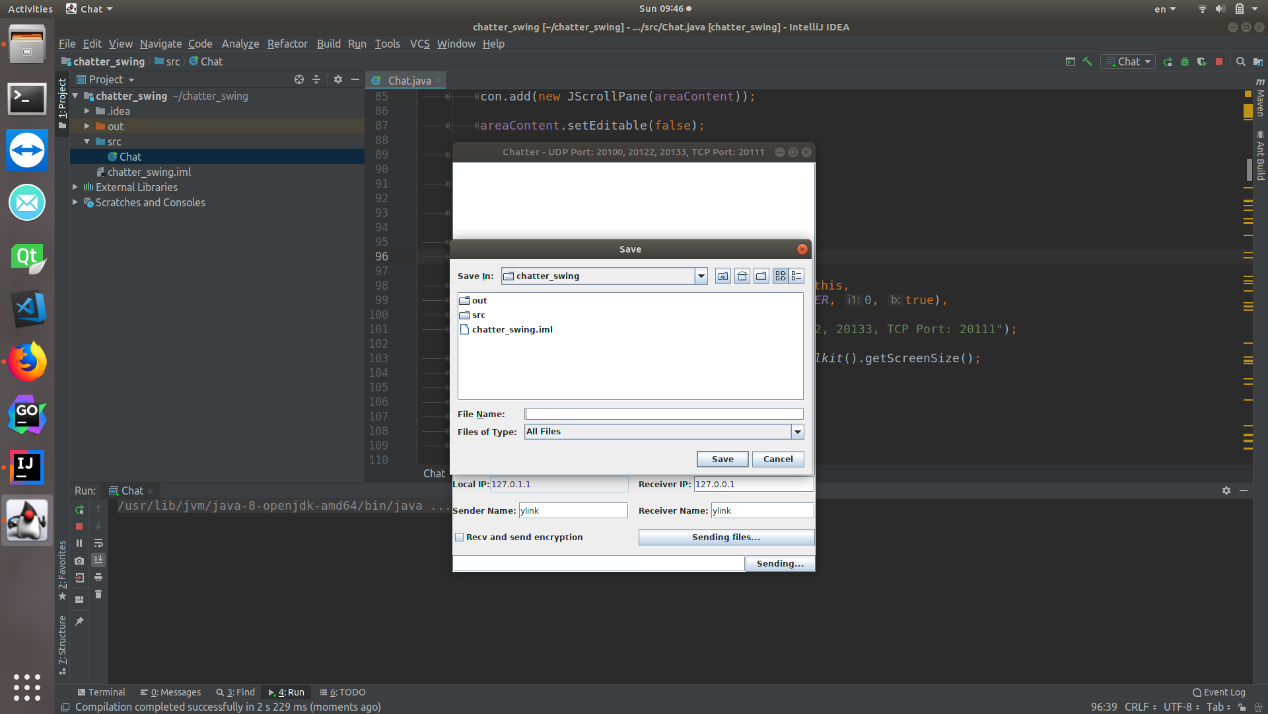
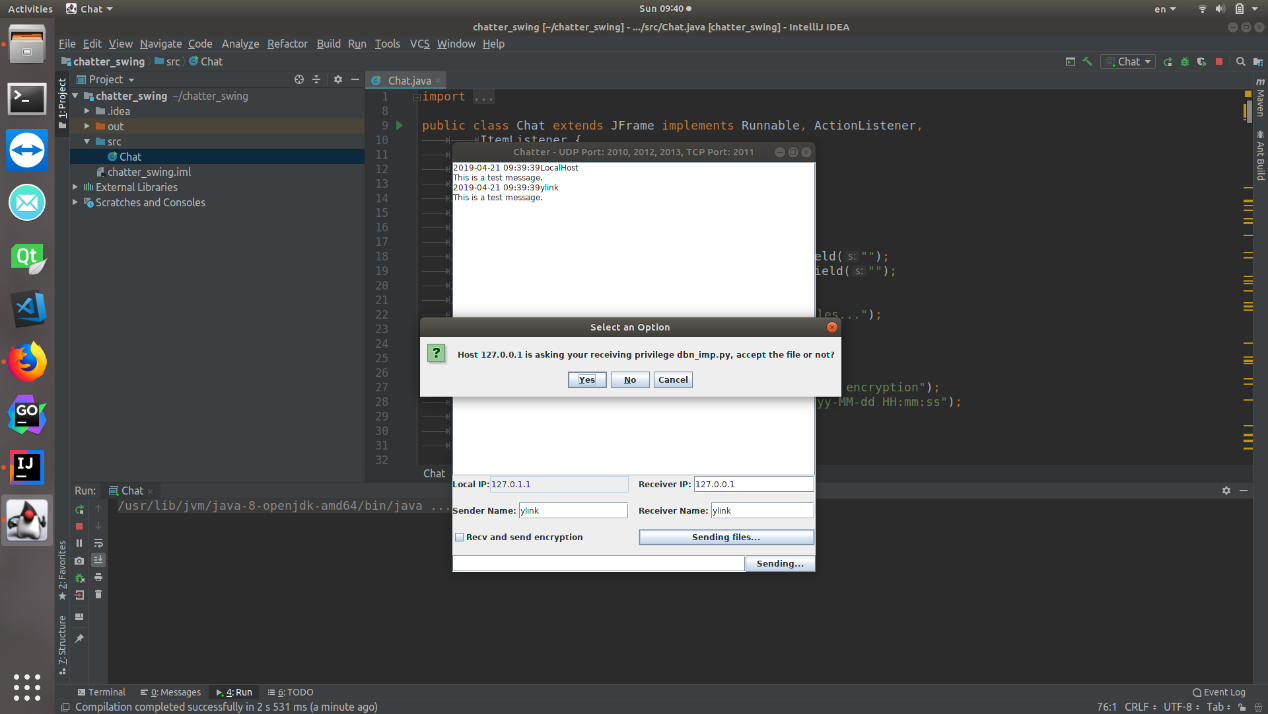
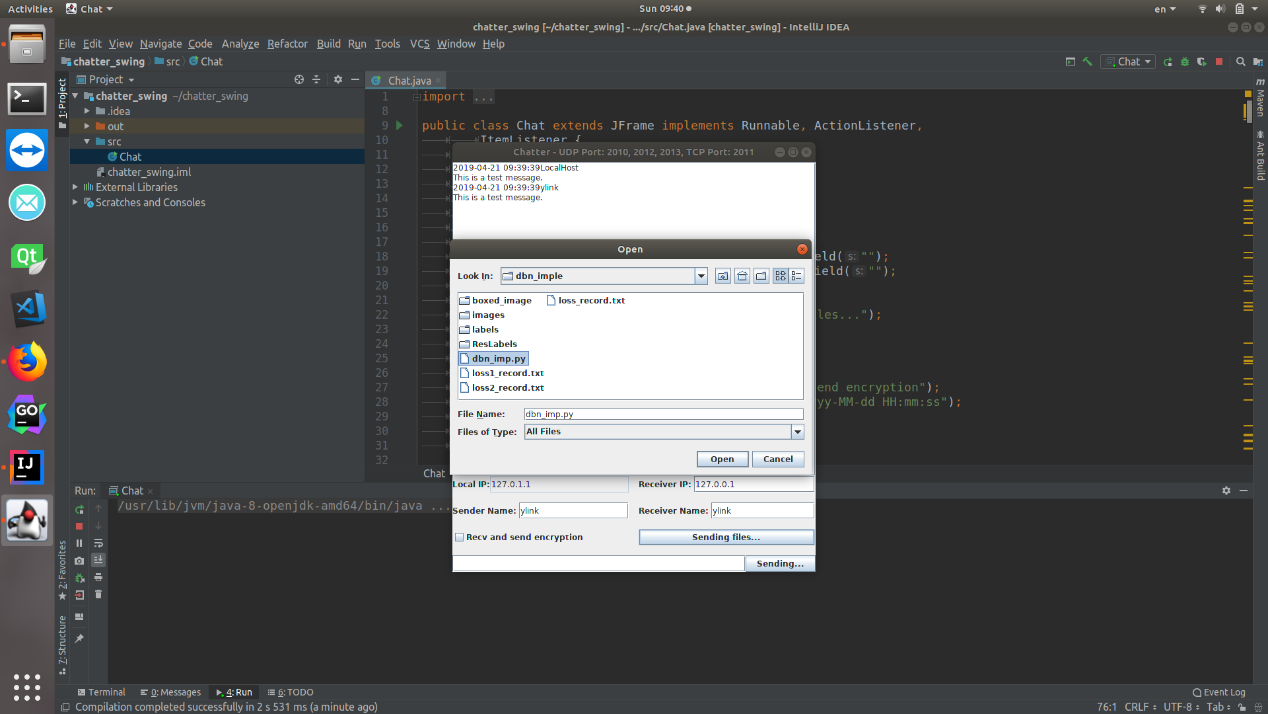
public void itemStateChanged(ItemEvent ie) {

this.isEncry = checkEncry.isSelected();

}

}

截图：



【小结或讨论】

通过本次实验，对java网络编程、java多线程编程的掌握得到了实质性提高。