**Copyfile\_Ui**

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.io.\*;

**import** java.sql.\*;

**import** javax.swing.\*;

**public** **class** UI\_copy\_file **extends** JFrame **implements** ActionListener {

**private** JButton bcmp=**null**;

**private** JButton bcp=**null**;

**private** JButton bshcnt;

**private** JTextField tresults;

**private** JLabel lfilesrc;

**private** JLabel lfiledes;

**private** JTextField tfilesrc;

**private** JTextField tfiledes;

**public** **boolean** compare\_file(String filesrc,String filedes) {

**try** (BufferedReader brsrc = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(filesrc)));

BufferedReader brdes = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(filedes)))) {

**while**(brsrc.ready() || brdes.ready()){

**if** (!brsrc.ready() || !brdes.ready()) **return** **false**;

String strsrc = brsrc.readLine();

String strdes= brdes.readLine();

System.***out***.println("??"+strsrc+">>"+strdes);

**if** (! strsrc.equals(strdes)) {

**return** **false**;

}

}

} **catch** (IOException exc) {

System.***out***.println("I/O Error: " + exc);

}

**return** **true**;

}

**public** **void** copy\_file(String filesrc,String filedes) {

String str;

**try** (BufferedReader br = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(filesrc)));

FileWriter fw = **new** FileWriter(filedes)) {

**while**(br.ready()){

str = br.readLine();

str = str + "\n"; // add newline

fw.write(str);

}

} **catch** (IOException exc) {

System.***out***.println("I/O Error: " + exc);

}

}

**public** **void** show\_content(String filename) {

**try** (BufferedReader br = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(filename)))) {

String str="";

**while**(br.ready()){

str =str+ br.readLine();

str = str + "\n"; // add newline

}

tresults.setText(str);

} **catch** (IOException exc) {

System.***out***.println("I/O Error: " + exc);

}

}

UI\_copy\_file(){

JFrame Jf=**new** JFrame("review on tap thoi cha noi");

Jf.setSize(1000,550);

Jf.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

Jf.setLayout(**null**);

Jf.setVisible(**true**);

lfilesrc=**new** JLabel("file source");

lfiledes=**new** JLabel("file destination");

tfilesrc = **new** JTextField(25);

tfiledes = **new** JTextField(25);

lfilesrc.setBounds(20,100,100,20);

tfilesrc.setBounds(100,100,100,20);

lfiledes.setBounds(200,100,100,20);

tfiledes.setBounds(300,100,100,20);

bcmp = **new** JButton("compare");

bcmp.setBounds(50, 150, 100,20);

bcp = **new** JButton("copy");

bcp.setBounds(170, 150, 100,20);

bshcnt = **new** JButton("show content");

bshcnt.setBounds(290, 150, 120,20);

tresults=**new** JTextField();

tresults.setBounds(450,100,400,300);

bcmp.addActionListener(**this**);

bcp.addActionListener(**this**);

bshcnt.addActionListener(**this**);

Jf.add(lfilesrc);

Jf.add(tfilesrc);

Jf.add(lfiledes);

Jf.add(tfiledes);

Jf.add(bcmp);

Jf.add(bcp);

Jf.add(bshcnt);

Jf.add(tresults);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

String comStr = e.getActionCommand();

**if** (comStr.equals("show content")) {

show\_content(tfilesrc.getText());

}

**else** **if** (comStr.equals("copy")) {

copy\_file(tfilesrc.getText(),tfiledes.getText());

}

**else** **if** (comStr.equals("compare")) {

**if** (compare\_file(tfilesrc.getText(),tfiledes.getText())) {

tresults.setText("Matched");

}

**else** tresults.setText("Not Matched");

}

//tresults.setText(comStr);

}

}

**import** java.awt.\*;

**import** javax.swing.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**public** **class** copy\_file {

copy\_file(){

**new** UI\_copy\_file();

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

SwingUtilities.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**new** copy\_file();

}

});

}

}

**database**

import java.awt.\*;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.table.\*;

public class SearchGUI extends JFrame implements ActionListener{

private JFrame add;

private JTable form;

private NhanVien nv;

private int cont=1;

private JTextField hoten;

private JTextField diachi;

private JTextField ngaysinh;

private JTextField dienthoai;

private JTextField chucvu;

private JTextField bangcap;

private JTextField bophan;

private DefaultTableModel MyTablemodel = new DefaultTableModel();

private Connection cnt;

SearchGUI(Connection conn){

cnt = conn;

JMenuBar jmb = new JMenuBar();

JMenu jmFile = new JMenu("Option");

JMenuItem jmiAdd = new JMenuItem("Add");

JMenuItem jmiSearch = new JMenuItem("Search");

jmFile.add(jmiAdd);

jmFile.add(jmiSearch);

jmb.add(jmFile);

JLabel jhoten=new JLabel("Họ tên");

JLabel jbophan=new JLabel("Bộ phận");

hoten = new JTextField(25);

bophan = new JTextField(25);

jhoten.setBounds(20,100,100,20);

hoten.setBounds(100,100,150,20);

jbophan.setBounds(260,100,100,20);

bophan.setBounds(320,100,150,20);

JButton b = new JButton("OK");

b.setBounds(200, 200, 60, 20);

b.addActionListener(this);

form = new JTable(MyTablemodel);

JScrollPane MyScrollPane= new JScrollPane(form);

MyTablemodel.addColumn("stt");

MyTablemodel.addColumn("Họ tên");

MyTablemodel.addColumn("Địa chỉ");

MyTablemodel.addColumn("Ngày sinh");

MyTablemodel.addColumn("Điện thoại");

MyTablemodel.addColumn("Chức vụ");

MyTablemodel.addColumn("Bằng cấp");

MyTablemodel.addColumn("Bộ phận");

// MyTablemodel.addRow(new Object[]{hoten, diachi, ngaysinh, dienthoai, chucvu, bangcap, bophan});

add = new JFrame();

add.setTitle("jTable Column Header Frame");

add.add(jhoten);

add.add(hoten);

add.add(jbophan);

add.add(bophan);

add.add(b);

add.setSize(500, 500);

add.add(MyScrollPane);

add.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

String comStr = e.getActionCommand();

System.out.print(comStr);

if (comStr.equals("OK")) {

String name = hoten.getText();

String bp = bophan.getText();

System.out.println(name+" "+bp);

Statement stmt=null;

try {

cnt.setAutoCommit(false);

stmt = cnt.createStatement();

String sql = "select \* from nhanvien where hoten='"+name+"' and "+"bophan='"+bp+"';";

System.out.println(sql);

ResultSet resultSet=stmt.executeQuery(sql);

while (resultSet.next()) {

String namer = resultSet.getString("hoten");

String dob = resultSet.getString("ngaysinh");

String bc = resultSet.getString("bangcap");

String cv = resultSet.getString("chucvu");

String ad = resultSet.getString("diachi");

String dt = resultSet.getString("dienthoai");

String bpr = resultSet.getString("bophan");

MyTablemodel.addRow(new Object[]{cont,namer, ad, dob, dt, cv, bc, bpr});

cont++;

}

cnt.commit();

} catch (SQLException e1) {

// TODO Auto-generated catch block

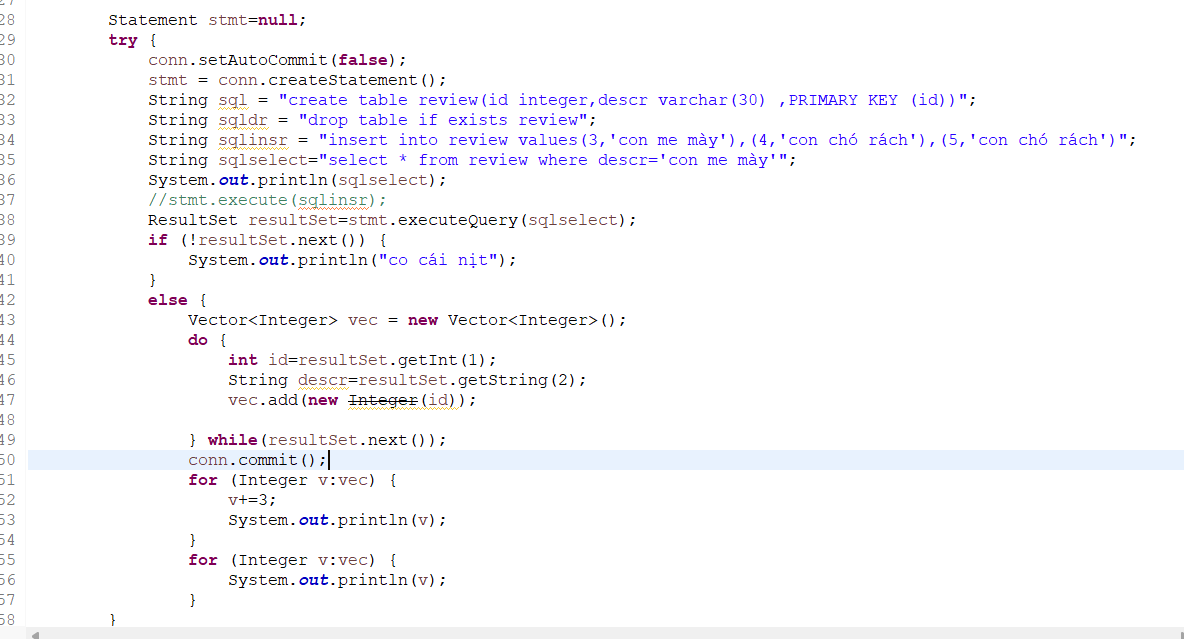
e1.printStackTrace();

}

}

}

}



NetWorking multithreads

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**package** network;

**import** java.net.\*;

**import** javax.net.ssl.\*;

**import** java.io.\*;

**import** java.util.\*;

**import** **static** java.lang.Thread.*currentThread*;

**class** Client\_Thread **implements** Runnable {

**private** String my\_name;

**private** Socket socket=**null**;

BufferedReader reader;

PrintWriter sender;

Client\_Thread (Socket thisSocket) **throws** IOException{

socket=thisSocket;

**try** {

reader=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));

sender=**new** PrintWriter(socket.getOutputStream());

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

};

// authenication username and password from client

String name=reader.readLine();

**if** (name!=**null**) {

my\_name=name;

System.***out***.println("Start chat for "+my\_name);

}

**else** {

System.***out***.println("Cannot receive name of Client");

}

// end

**try** {

**for** (Map.Entry me : Server.*Name2Socket*.entrySet()) {

System.***out***.println("Key: "+me.getKey());

PrintWriter sender\_User=**null**;

sender\_User=**new** PrintWriter(((Socket) me.getValue()).getOutputStream());

sender\_User.println(my\_name+" has joined");

sender\_User.flush();

}

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

Server.*Name2Socket*.put(my\_name, socket);

//

// sender.println(my\_name);

// sender.flush();

}

**public** **void** run()

{

**try** {

**while** (**true**) {

String raw\_text=**null**,message=**null**,tag=**null**;

raw\_text = reader.readLine();

**if** (raw\_text!=**null**) {

tag=raw\_text.substring(0,1);

**if** (tag.equals("M")) { // Me message

System.***out***.println("raw text "+raw\_text);

message=raw\_text.substring(2,raw\_text.length());

}

**else** **if** (tag.equals("Q")) { // Qi Quit

message=my\_name+" has left";

}

}

**if** (message!=**null**) {

**for** (Map.Entry me : Server.*Name2Socket*.entrySet()) {

**if** (!(my\_name.equals(me.getKey()))){

System.***out***.println("send from "+my\_name+" to "+me.getKey());

PrintWriter sender\_User=**null**;

sender\_User=**new** PrintWriter(((Socket) me.getValue()).getOutputStream());

**if** (!tag.equals("Q")) sender\_User.println(my\_name+": "+message);

**else** sender\_User.println(message);

sender\_User.flush();

}

}

}

**if** (tag.equals("Q")) {

Server.*Name2Socket*.remove(my\_name);

*currentThread*().interrupt();

**return**;

}

}

}

**catch** (IOException e) {

// **TODO** Auto-generated catch blocka

e.printStackTrace();

}

}

}

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Graphical user interface, text

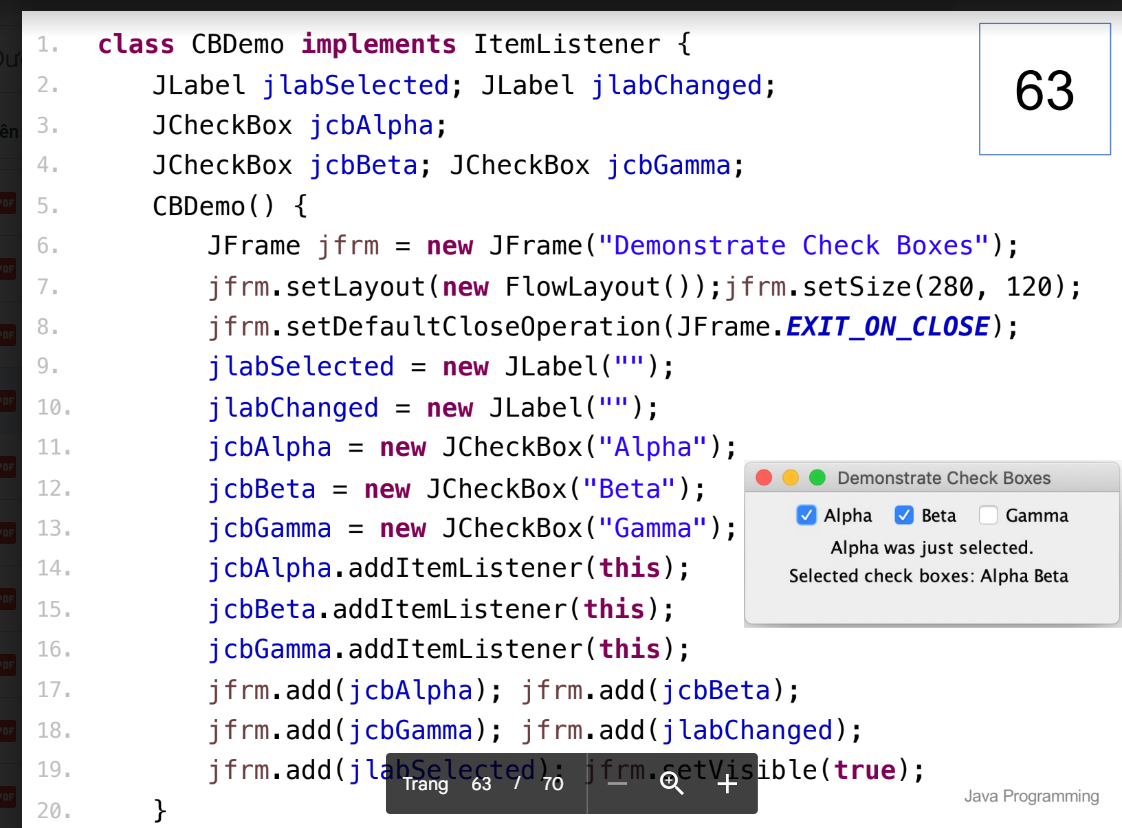
Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, text, table

Description automatically generated with medium confidence



Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generatedText

Description automatically generated