**A**

**PROJECT REPORT**

**ON**

**Food Delivery SYSTEM**

for

[software engineering](https://fanyi.so.com/?src=onebox" \l "software engineering" \t "https://www.so.com/_blank)（NIIT）project

**Submited**

**By**

（Fatty delivery|牛聪 Daniel，闫冰洁Jayce，滕畅 Krystal，王楠 Bonnie）

R8J9400097，R8J9400117，R8J9400123，R8J9400106|2019.5.1-2019.5.23

**Internal Guide External Guide**

**Mr.Ashok Khandelwal Mr.Decai Liu**

**Head of NIIT teacher from Hainu**

**in Hainu University University**





**Computer and Cyberspace Academy security in Hainu University**

PROJECT INDEX

|  |  |  |
| --- | --- | --- |
| S no. | Particulars | Page no. |
| 1 | About the project,requirement,SDLC,Analysis | 1-2 |
| 2 | About the modules:  Login module,  Bussiness module,  Customer module | 2-3 |
| 3 | Use case Diagram | 3-4 |
| 4 | DFD of Data Flow Diagram | 5 |
| 5 | DFD of Login module | 6 |
| 6 | Bussiness module DFD | 6 |
| 7 | User module DFD | 7 |
| 8 | Testing | 7 |
| 9 | COST ESTIMATION | 7-8 |
| 10 | Code and corresponding screenshot of the project | 8-102 |
|  |  |  |

TEAM DESCRIPTION & OVERVIEW

Fatty delivery,people like their names,we are a Group of people who are called fatty.So we want to figure out a way to develop fatty’s common life.We may not the smartest team through all the thirty team,but we certainly are the hardest team in Hainu University !

**CERTIFICATE**

This is to certify that this report,titled Fatty delivery,embodies the original work done by Niu Cong，Yan Bingjie,Wang Nan and Teng Chang in partial fulfillment of their course requirement at NIIT.

Coordinator:Mr.Ashok and Liu Decai

20th May 2019

**LETTER OF ACKNOWLEDGEMENT**

To dear Ashok and Liu Decai teachers:

After our untiring efforts for this period of time, our project is nearing completion. From the beginning, there is no clue, from the knowledge to the present, the approach is perfect, and we have a lot of energy and strength. We have cried, we have laughed, we have been silent. Of course, our results are outstanding, and we have gained a lot.

Every minute of our project from the beginning of the project to the end of the project, our dear teachers have spent the same time with us, patiently answering each of our questions and focusing on imparting professional knowledge to us. We really appreciate our two instructors, and all of our Fatty delivery members sincerely thank Ashok and Liu Decai!

I believe that we will get enough knowledge and make better Internet projects under your guidance. We will always be your loyal fans!

We love you!

Fatty delivery Group

**PROJECT REPORT**

**Food Delivery SYSTEM**

Nowadays，delivery has been more and more popular in our common life.To met our increasing demand towards delivery,we need a good Takeaway ordering system.The system is composed by login interface, custommer interface and Bussiness interface.We have carefully designed the interface, selected the images that fit the theme, and optimized the functions and code of each program. When entering the login screen, we have two options - login or registration, you need to choose the customer and merchant options.

**Software Requirements**

* jdk1.6.0\_03＆jre1.6.0\_03
* NetBeans IDE 7.1
* Web Logic Server
* MySQL 5.5 Command Line Client\workbench

A large number of merchant and customer information needs to be entered and stored, and can be searched anytime, anywhere. This information often involves a variety of orders and merchandise, and our take-away system can efficiently input this information into the computer port, package and load this important information, and electronically implement all processes semi-automated. Our takeaway system connects java to MySQL and finally saves the input information in a MySQL table.Fatty Delivery is widely used to provide continuous convenience to both parties.

**SDLC of the Project**

Our project is a web application and going to be developed using java technology.Similar to all other software projects, the project also has its own software development.We will talk about it in the following text.

**Requirement Analysis**

Before the project starts, the first thing to consider is the requirements. The requirements for new system fills can be referred to as demand analysis by properly analyzing the ex. system for its problems. Requirement analysis is the most important part to be considered before developing any project as be must be aware of the objective s that be wish to achieve by making this project. This analysis identifies the problems and possible solutions faced by users or customers. Program. We can use some questionnaires to find out what the customer is facing.

We need to collect a large number of different types of user information, analyze and process this information, and summarize the main and secondary needs of users. After drawing the conclusion, we will add relevant design to the program to suit the user's usage habits.

**Software Design**

Software design is a solution for making the right program - "How to create a new system." This important phase consists of several steps. It provides the deep understanding and procedural details needed to implement the system proposed in the feasibility study.

There are three distinct types of activities:

* External design：choosing kinds of pictures，Layer cutting the Exterior design.
* Architecture design:Face the user's needs for the primary design, and then gradually improve the structure.
* Detailed design :Firstly,Wewrote the program code that can perform the basic functions, and slowly improved the structure and efficiency of the code.

**Login module**

When we complete the registration or log in to the customer interface, we can see two interfaces. An interface for ordering and confirming and changing personal information; an interface for viewing your own order and modifying order information. We can choose the merchants we like in the customer interface and make purchases based on the sample images given by the food.

The development of this module includes writing code for the following files：

* LoginFrame.java
* Register.java
* Main.java
* textClass.java
* DBConnect.java
* PJFont.java
* User.java

**User module**

We can also change the demand information of the order in the order interface. If you want to delay the delivery time of the take-out, you can modify the scheduled delivery time in the order interface. The merchant will receive the changed order information and notify the rider in the first time. Change the corresponding itinerary, of course, you can also contact the merchant to return the order.

This part of function is implemented by the code:

* Main.java
* textClass.java
* User.java
* Commodity.java
* PJFont.java
* DBConnect.java
* CommodityList.java
* CommodityPanel.java
* CommoditySelectPanel.java
* CustomerFrame.java
* CustomerInformationPage.java
* CustomerOrderList.java
* OrderDetailFrame.java

**Bussiness module**

After selecting the merchant option to enter the merchant interface, we can see the product menu and order menu, as well as the personal information interface. In the product interface, you can add and remove goods, and customize the price and name of the product. As a merchant, we can also view the specific information of the bill and receive the information changed by the user in the order interface.

* Main.java
* textClass.java
* User.java
* Commodity.java
* DBConnect.java
* PJFont.java
* AddcommodityFrame.java
* BussinessCommodityList.java
* BussinessFrame.java
* BussinessInformationPage.java
* BussinessOrderList.java

**Database Design**

The entire project is actually an information management system that interacts with a database with multiple tables. The table and its column names and data types are as follows.

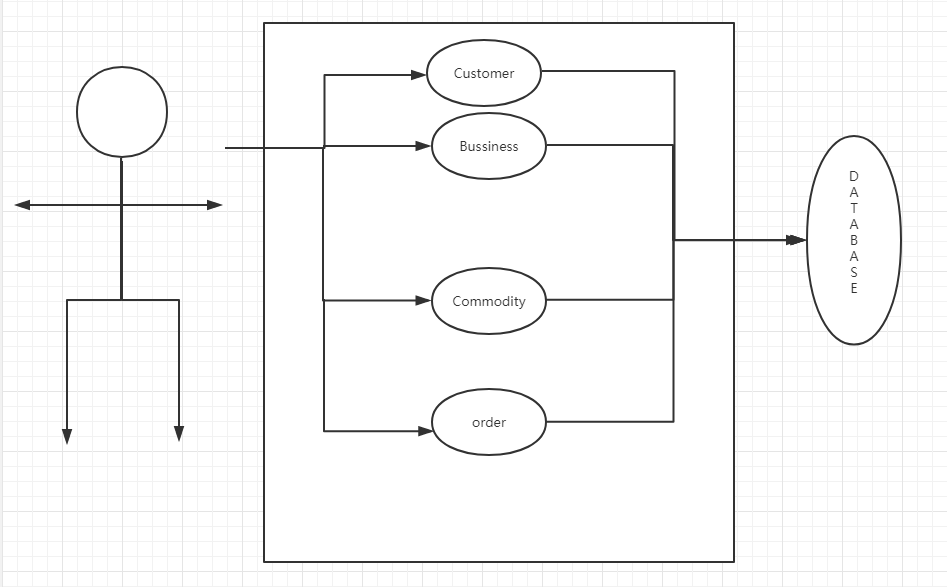
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customertable | | | | |  |
| **Field Name** | **Data Type** | **Size** | **Primary Key** | **Null** | **Foreigh Key** |
| username | Varchar | 20 | Y | N |  |
| password | Varchar | 20 |  | N |  |
| gender | char | 6 |  |  |  |
| phone | Varchar | 20 |  | N |  |
| email | Varchar |  |  |  |  |
| dob | Date |  |  |  |  |
| addr | Varchar | 50 |  | N |  |
| regtime | datetime |  |  | N |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BussinessTable | | | | |  |
| **Field Name** | **Data Type** | **Size** | **Primary Key** | **Null** | **Foreigh Key** |
| username | Varchar | 20 | Y | N |  |
| password | Varchar | 20 |  | N |  |
| gender | char | 6 |  |  |  |
| phone | Varchar | 20 |  | N |  |
| email | Varchar | 30 |  |  |  |
| dob | date |  |  |  |  |
| addr | Varchar | 50 |  | N |  |
| regtime | datetime |  |  | N |  |

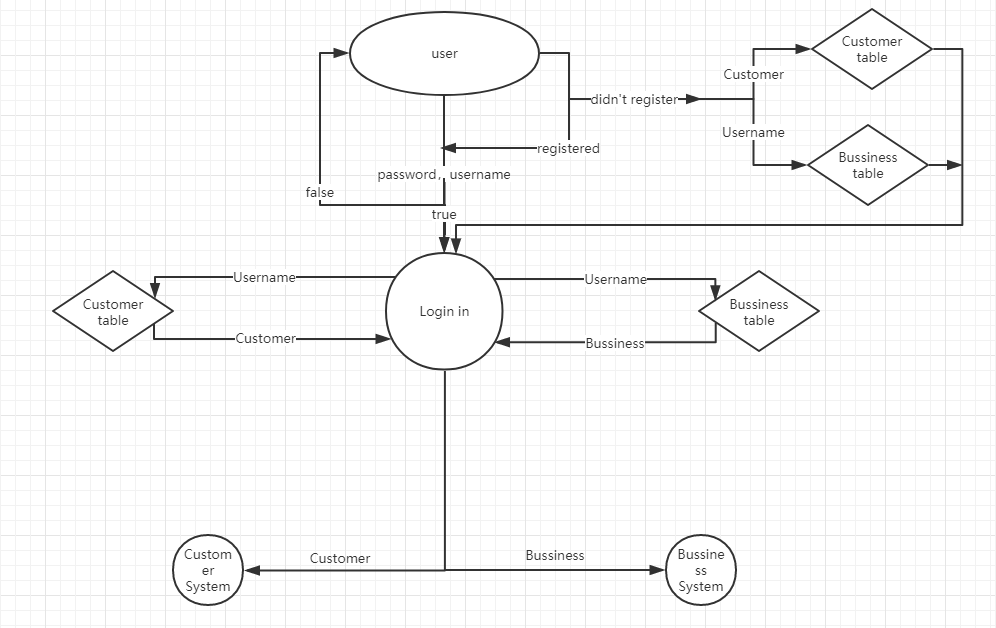
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CommodityTable | | | | |  |
| **Field Name** | **Data Type** | **Size** | **Primary Key** | **Null** | **Foreigh Key** |
| id | Varchar | 50 | Y | N |  |
| cost | double |  |  | N |  |
| bussinessid | Varchar | 20 |  | N | Bussinesstable(username) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OrderTable | | | | |  |
| **Field Name** | **Data Type** | **Size** | **Primary Key** | **Null** | **Foreigh Key** |
| addr | Varchar | 50 |  | N |  |
| amount | double |  |  | N |  |
| bussinessid | Varchar | 20 |  | N | BussinessTable(Username) |
| customerid | Varchar | 20 |  | N | CustomerTable(Username) |
| commodityid | Varchar | 50 |  | N |  |
| datetime | datetime |  |  | N |  |
| status | Varchar | 20 |  | N |  |
| phone | Varchar | 20 |  | N |  |
|  |  |  |  |  |  |
| orderid | bigint |  | Y | N |  |

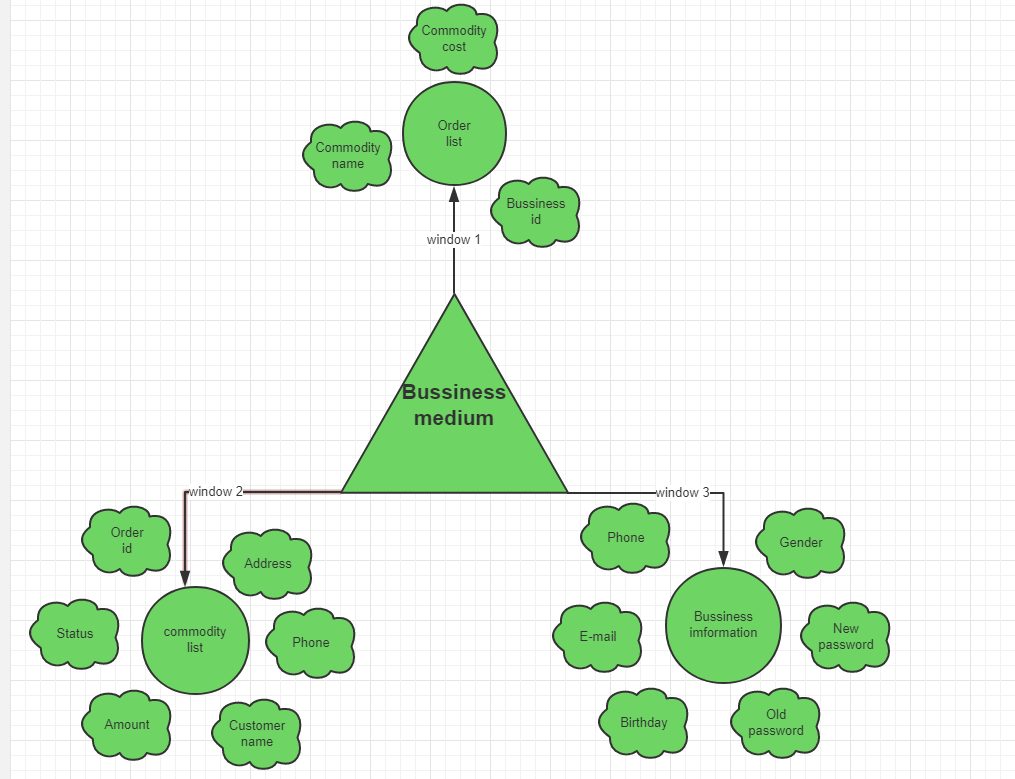
**Use Case diagram**



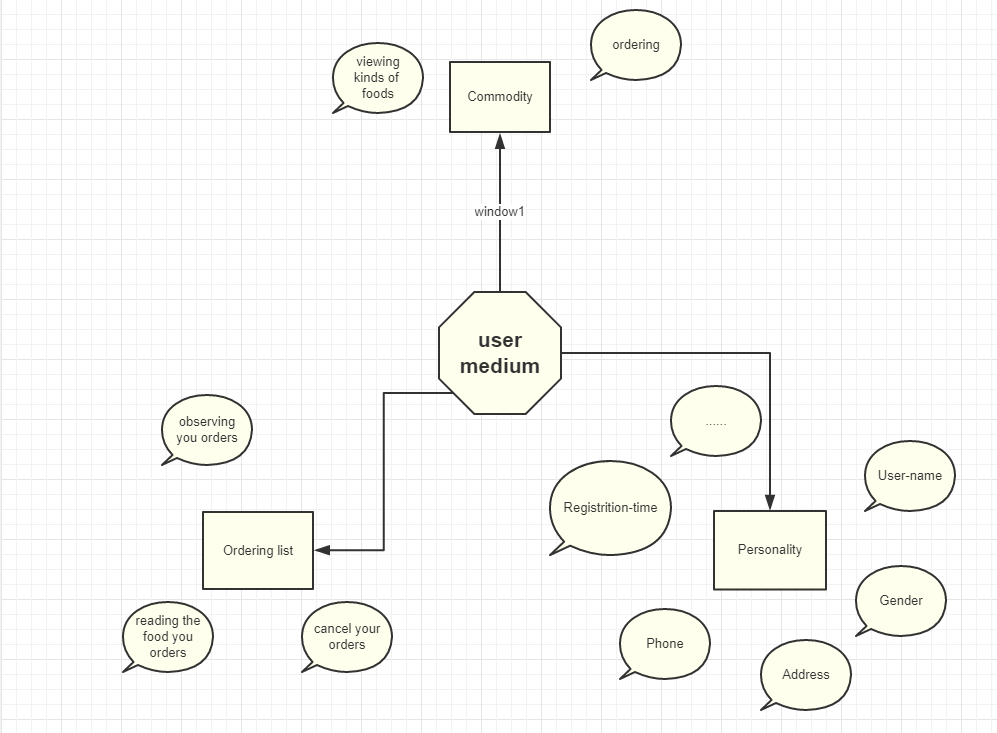
**DFD of Data Flow Diagram:**



**Login module DFD:**



**Bussiness module DFD：**



**User module DFD：**

**Testing**

We first unit test each class to test whether each method of each class can correctly implement the corresponding function; then through the integration test, the various classes are connected to each other, and the test is turned into a whole.

The test method is mainly black box test, mainly through input and obtained results, such as using System.out to output the value of the intermediate variable after the execution of the corresponding step and the content displayed on the form to determine whether the corresponding function is correctly implemented.

We used a lot of test data in the test process, divided the test data into several equivalence classes according to the software writing content, and selected some test data for the equivalent class to conduct relevant tests. For example, when we insert images into products, we are divided into the following equivalence classes: no image, incorrect image format, large image size anding test results.

**COST ESTIMATION**

After more than 20 days, our project development is nearing completion. In so many days of hard work, our mind and ability have been greatly improved, and we have also gained a lot of expertise we need. Let's take a look at the time we spent on making projects for so many days. See the chart below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Work Done By** | **Work task** | **Starting Date** | **Ending Date** | **Duration(day)** |
| Yan Bingjie | Calling class＆Bussiness module | 1th May 19 | 16th May 19 | 15 |
| Wang Nan | Customer module | 1th May 19 | 19th May 19 | 18 |
| Teng Chang | Login module | 1th May 19 | 17th May 19 | 16 |
| Niu Cong | Database＆Report  ·ppt | 1th May 19 | 22th May 19 | 21 |

This software has been developed in 21 days. We work on this project about 4.5 hours a day（except me 2 hours）.

The total hours = 21\*2+49\*4.5= 262.5

Estimated the working cost of manpower is $1.2/Hour.

Then the total cost of the project is = 262.5\*1.2\*7 = 2205

So the Estimated Total cost of the Project is Two Thousand, Two Hundred and 5 Yuan only.

**Fattydelivery**

**Main.java**

package fattydelivery;

import fattydelivery.lib.\*;

import fattydelivery.log.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Administrator

\*/

public class Main {

public static void main(String[] args) {

new LoginFrame();

// try {

//

// // 添加客户或商户

// User u = new User();

// u.setUsername("beiyu");

// u.setPassword("beiyu");

// u.setRegTime();

// u.setPhone("15666676912");

// u.setAddr("Haikou, Haihan");

// u.setDob("2018-05-04"); // 可以为空

// u.setEmail("729320011@qq.com"); // 可以为空

// u.setGender("Male"); // 可以为空

// u.setUsertype(2);// 1为用户， 2为商户

// if(u.addToDB()==0) { // 判断用户名是否被注册过并添加到数据库中

// System.out.println("username has exist.");

// } else {

// System.out.println("Success"); // 添加成功

// }

///\*

// // 添加商品

// Commodity c = new Commodity();

// c.setBussinessId("beiyu");

// c.setCost(23.55);

// c.setId("food1");

// if(c.addToDB()==0) { // 判断id是否冲突并添加到数据库中

// System.out.println("id has exist.");

// } else {

// System.out.println("Success"); // 添加成功

// }

//\*/

//

// // 添加订单

// Order o = new Order(); // orderid自动生成 不必set setTime()已经调用过也可以不必调用

// o.setAddress(u.getAddr());

// o.setAmount(23.55\*5);

// o.setBussinessId(u.getUsername());

// o.setCustomId(u.getUsername());

// o.setPhone(u.getPhone());

// // 一个商品时

// o.setCommodityId("food1, food1, food1");

// // 多个商品时

//// o.setCommodityId("food1，food2");

// //或者

//// String[] s= new String[5];

//// s[0]="food1"; s[1]="food1"; s[2]="food1"; s[3]="food1"; s[4]="food1";

//// o.setCommodityId(s);

// o.setStatus("Preparing"); // 准备中

// o.setStatus("Delivering"); // 递送中

// o.setStatus("Delivered"); // 已送达

// o.addToDB();

//

// } catch (Exception ex) {

// Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);

// }

}

}

**textClass.java**

package fattydelivery;

import fattydelivery.lib.\*;

/\*\*

\*

\* @author Administrator

\*/

public class textClass {

public static void main(String[] args) {

User u = new User();

u.setUsername("jiachangcai");

u.setPassword("88888888");

u.setDob("1998-12-24");

u.setAddr("People Street, Haikou, Hainan, China");

u.setGender("male");

u.setEmail("8888@qq.com");

u.setPhone("19955668877");

u.setRegTime();

u.setUsertype(2);

u.addToDB();

u.setUsername("niupai");

u.setPassword("88888888");

u.setDob("1999-1-24");

u.setAddr("Haidiandao, Haikou, Hainan, China");

u.setGender("male");

u.setEmail("6666@qq.com");

u.setPhone("11223344567");

u.setRegTime();

u.setUsertype(2);

u.addToDB();

u.setUsername("pisa");

u.setPassword("88888888");

u.setDob("2000-1-24");

u.setAddr("Renmentianqiao, Haikou, Hainan, China");

u.setGender("female");

u.setEmail("wwww6@qq.com");

u.setPhone("11223344568");

u.setRegTime();

u.setUsertype(2);

u.addToDB();

u.setUsername("guke");

u.setPassword("88888888");

u.setDob("1998-11-24");

u.setAddr("58 th People Street, Haikou, Hainan, China");

u.setGender("female");

u.setEmail("wwww@qq.com");

u.setPhone("19955668977");

u.setRegTime();

u.setUsertype(1);

u.addToDB();

Commodity c = new Commodity();

c.setImg("foodimg\\galizhupaifan.jpg");

c.setBussinessId("jiachangcai");

c.setId("galizhupaifan");

c.setCost(12);

c.addToDB();

c.setImg("foodimg\\huangmenjimifan.jpg");

c.setBussinessId("jiachangcai");

c.setId("huangmenjimifan");

c.setCost(15);

c.addToDB();

c.setImg("foodimg\\qiezibao.jpg");

c.setBussinessId("jiachangcai");

c.setId("qiezibao");

c.setCost(12);

c.addToDB();

c.setImg("foodimg\\huoyanniupai.jpg");

c.setBussinessId("niupai");

c.setId("huoyanniupai");

c.setCost(38);

c.addToDB();

c.setImg("foodimg\\xinaoerliangkaoroupisa.jpg");

c.setBussinessId("pisa");

c.setId("xinaoerliangkaoroupisa");

c.setCost(22);

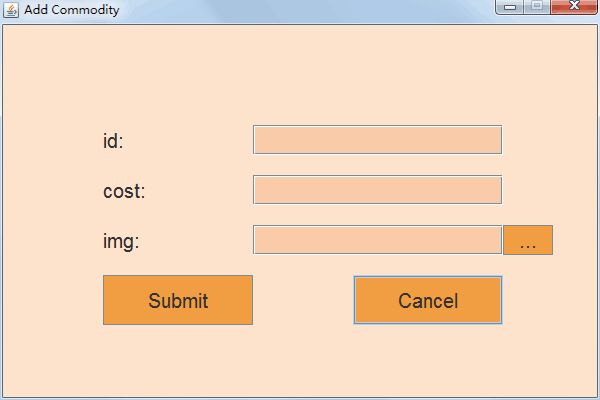
c.addToDB();

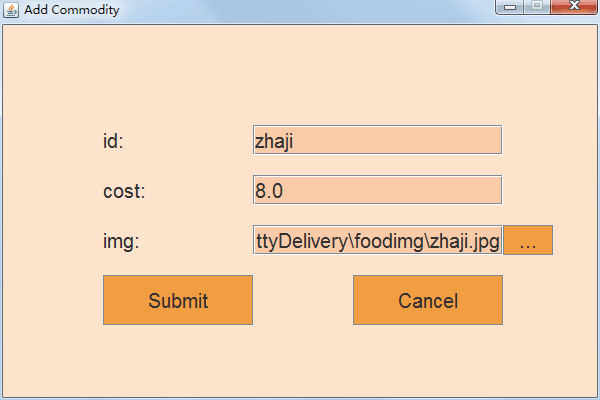
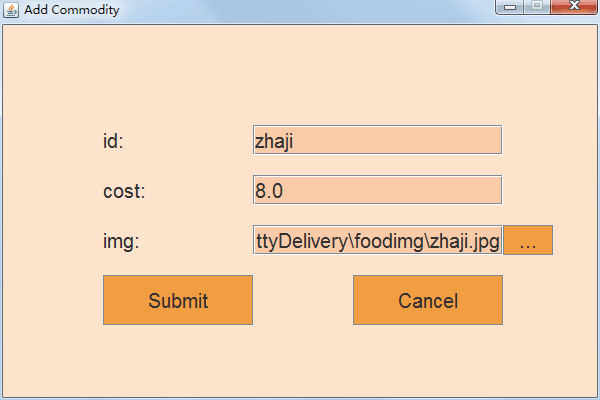
}

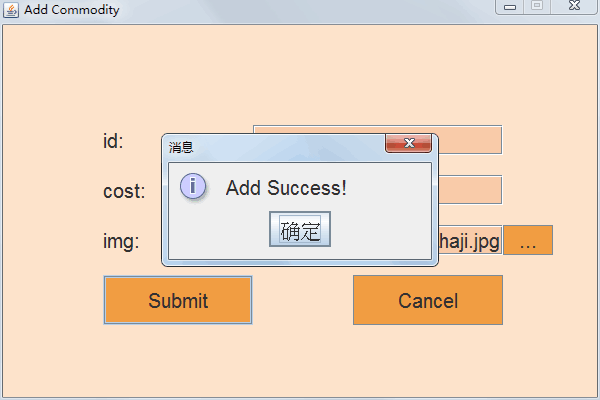
}

**fattydelivery.bussiness**

**AddcommodityFrame.java**

****

****

****

package fattydelivery.bussiness;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

/\*\*

\*

\* @author Administrator

\*/

public class AddCommodityFrame extends JFrame implements ActionListener {

private Commodity com;

private JPanel JP;

private JLabel JLid, JLcost, JLimg;

private JTextField JTFid, JTFcost, JTFimg;

private JButton JBsubmit, JBcancel, JBupload;

private String filePath;

public AddCommodityFrame(User u) {

super("Add Commodity");

com = new Commodity();

com.setBussinessId(u.getUsername());

Build();

}

private void Build() {

this.setResizable(false);

this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

this.setLocationRelativeTo(null);

this.setSize(600, 400);

JP = new JPanel(null);

JP.setFont(PJFont.ft);

JLid = new JLabel("id:");

JLcost = new JLabel("cost:");

JTFid = new JTextField(30);

JTFcost = new JTextField(30);

JLimg = new JLabel("img:");

JTFimg = new JTextField(30);

JLid.setFont(PJFont.ft);

JLcost.setFont(PJFont.ft);

JTFid.setFont(PJFont.ft);

JTFcost.setFont(PJFont.ft);

JLimg.setFont(PJFont.ft);

JTFimg.setFont(PJFont.ft);

JLid.setBounds(100, 100, 50, 30);

JTFid.setBounds(250, 100, 250, 30);

JLcost.setBounds(100, 150, 50, 30);

JTFcost.setBounds(250, 150, 250, 30);

JLimg.setBounds(100, 200, 50, 30);

JTFimg.setBounds(250, 200, 250, 30);

JBsubmit = new JButton("Submit");

JBcancel = new JButton("Cancel");

JBupload = new JButton("...");

JBsubmit.addActionListener(this);

JBcancel.addActionListener(this);

JBupload.addActionListener(this);

JBsubmit.setBounds(100, 250, 150, 50);

JBcancel.setBounds(350, 250, 150, 50);

JBupload.setBounds(500, 200, 50, 30);

JBsubmit.setFont(PJFont.ft);

JBcancel.setFont(PJFont.ft);

JBupload.setFont(PJFont.ft);

JP.add(JLid);

JP.add(JTFid);

JP.add(JLcost);

JP.add(JTFcost);

JP.add(JLimg);

JP.add(JTFimg);

JP.add(JBupload);

JP.add(JBsubmit);

JP.add(JBcancel);

this.add(JP);

this.setVisible(true);

JBsubmit.setBackground(new Color(241, 156, 67));

JBcancel.setBackground(new Color(241, 156, 67));

JBupload.setBackground(new Color(241, 156, 67));

JTFid.setBackground(new Color(249, 203, 168));

JTFcost.setBackground(new Color(249, 203, 168));

JTFimg.setBackground(new Color(249, 203, 168));

JP.setBackground(new Color(252, 226, 203));

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBsubmit) {

String str;

String regex;

Pattern pattern;

Matcher matcher;

str = JTFid.getText();

regex = "^[a-zA-Z0-9\_]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "Commodity name must not be special character.");

return;

}

str = JTFcost.getText();

regex = "^[0-9]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

Matcher matcher1 = Pattern.compile("^[0-9]+.[0-9]{0,2}$").matcher(str);

if (!matcher.matches() && !matcher1.matches()) {

JOptionPane.showMessageDialog(this, "Cost must not be special character.");

return;

}

com.setId(JTFid.getText());

com.setCost(Double.parseDouble(JTFcost.getText()));

if (filePath == null) {

com.setImg("img\\fatty.jpg");

} else {

com.setImg(filePath);

}

// System.out.println(com);

if (com.addToDB() != 0) {

JOptionPane.showMessageDialog(this, "Add Success!");

this.dispose();

} else {

JOptionPane.showMessageDialog(this, "id already exists.");

}

} else if (e.getSource() == JBcancel) {

this.dispose();

} else if (e.getSource() == JBupload) {

JFileChooser chooser = new JFileChooser("/C:/");

FileNameExtensionFilter filter = new FileNameExtensionFilter("Image files", "jpg", "jpeg", "gif", "png");

chooser.setFileFilter(filter);

int result = chooser.showOpenDialog(this);

String uploadPath = null;

String selectedPath = null;

String fileName = null;

if (result == JFileChooser.APPROVE\_OPTION) {

File selectedFile = chooser.getSelectedFile();

selectedPath = selectedFile.getPath();

fileName = selectedFile.getName();

String ext = fileName.substring(fileName.lastIndexOf(".") + 1);

// System.out.println(ext);

// System.out.println(selectedPath);

// System.out.println(selectedFile.getName());

if (!"jpg".equalsIgnoreCase(ext)

&& !"jpeg".equalsIgnoreCase(ext)

&& !"gif".equalsIgnoreCase(ext)

&& !"png".equalsIgnoreCase(ext)) {

JOptionPane.showMessageDialog(this, "only image file you can choose here.");

return;

}

// System.out.println(selectedFile.length());

if (selectedFile.length() > 1024 \* 1024) {

JOptionPane.showMessageDialog(this, "image size must smaller than 1MB.");

return;

}

filePath = selectedPath;

JTFimg.setText(filePath);

}

}

}

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

// User u = new User();

// u.setUsername("beiyu");

// u.setUsertype(2);

// u.getFromDB();

// new AddCommodityFrame(u);

// }

}

**BussinessCommodityList.java**

****

****

package fattydelivery.bussiness;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

/\*\*

\*

\* @author Administrator

\*/

public class AddCommodityFrame extends JFrame implements ActionListener {

private Commodity com;

private JPanel JP;

private JLabel JLid, JLcost, JLimg;

private JTextField JTFid, JTFcost, JTFimg;

private JButton JBsubmit, JBcancel, JBupload;

private String filePath;

public AddCommodityFrame(User u) {

super("Add Commodity");

com = new Commodity();

com.setBussinessId(u.getUsername());

Build();

}

private void Build() {

this.setResizable(false);

this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

this.setLocationRelativeTo(null);

this.setSize(600, 400);

JP = new JPanel(null);

JP.setFont(PJFont.ft);

JLid = new JLabel("id:");

JLcost = new JLabel("cost:");

JTFid = new JTextField(30);

JTFcost = new JTextField(30);

JLimg = new JLabel("img:");

JTFimg = new JTextField(30);

JLid.setFont(PJFont.ft);

JLcost.setFont(PJFont.ft);

JTFid.setFont(PJFont.ft);

JTFcost.setFont(PJFont.ft);

JLimg.setFont(PJFont.ft);

JTFimg.setFont(PJFont.ft);

JLid.setBounds(100, 100, 50, 30);

JTFid.setBounds(250, 100, 250, 30);

JLcost.setBounds(100, 150, 50, 30);

JTFcost.setBounds(250, 150, 250, 30);

JLimg.setBounds(100, 200, 50, 30);

JTFimg.setBounds(250, 200, 250, 30);

JBsubmit = new JButton("Submit");

JBcancel = new JButton("Cancel");

JBupload = new JButton("...");

JBsubmit.addActionListener(this);

JBcancel.addActionListener(this);

JBupload.addActionListener(this);

JBsubmit.setBounds(100, 250, 150, 50);

JBcancel.setBounds(350, 250, 150, 50);

JBupload.setBounds(500, 200, 50, 30);

JBsubmit.setFont(PJFont.ft);

JBcancel.setFont(PJFont.ft);

JBupload.setFont(PJFont.ft);

JP.add(JLid);

JP.add(JTFid);

JP.add(JLcost);

JP.add(JTFcost);

JP.add(JLimg);

JP.add(JTFimg);

JP.add(JBupload);

JP.add(JBsubmit);

JP.add(JBcancel);

this.add(JP);

this.setVisible(true);

JBsubmit.setBackground(new Color(241, 156, 67));

JBcancel.setBackground(new Color(241, 156, 67));

JBupload.setBackground(new Color(241, 156, 67));

JTFid.setBackground(new Color(249, 203, 168));

JTFcost.setBackground(new Color(249, 203, 168));

JTFimg.setBackground(new Color(249, 203, 168));

JP.setBackground(new Color(252, 226, 203));

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBsubmit) {

String str;

String regex;

Pattern pattern;

Matcher matcher;

str = JTFid.getText();

regex = "^[a-zA-Z0-9\_]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "Commodity name must not be special character.");

return;

}

str = JTFcost.getText();

regex = "^[0-9]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

Matcher matcher1 = Pattern.compile("^[0-9]+.[0-9]{0,2}$").matcher(str);

if (!matcher.matches() && !matcher1.matches()) {

JOptionPane.showMessageDialog(this, "Cost must not be special character.");

return;

}

com.setId(JTFid.getText());

com.setCost(Double.parseDouble(JTFcost.getText()));

if (filePath == null) {

com.setImg("img\\fatty.jpg");

} else {

com.setImg(filePath);

}

// System.out.println(com);

if (com.addToDB() != 0) {

JOptionPane.showMessageDialog(this, "Add Success!");

this.dispose();

} else {

JOptionPane.showMessageDialog(this, "id already exists.");

}

} else if (e.getSource() == JBcancel) {

this.dispose();

} else if (e.getSource() == JBupload) {

JFileChooser chooser = new JFileChooser("/C:/");

FileNameExtensionFilter filter = new FileNameExtensionFilter("Image files", "jpg", "jpeg", "gif", "png");

chooser.setFileFilter(filter);

int result = chooser.showOpenDialog(this);

String uploadPath = null;

String selectedPath = null;

String fileName = null;

if (result == JFileChooser.APPROVE\_OPTION) {

File selectedFile = chooser.getSelectedFile();

selectedPath = selectedFile.getPath();

fileName = selectedFile.getName();

String ext = fileName.substring(fileName.lastIndexOf(".") + 1);

// System.out.println(ext);

// System.out.println(selectedPath);

// System.out.println(selectedFile.getName());

if (!"jpg".equalsIgnoreCase(ext)

&& !"jpeg".equalsIgnoreCase(ext)

&& !"gif".equalsIgnoreCase(ext)

&& !"png".equalsIgnoreCase(ext)) {

JOptionPane.showMessageDialog(this, "only image file you can choose here.");

return;

}

// System.out.println(selectedFile.length());

if (selectedFile.length() > 1024 \* 1024) {

JOptionPane.showMessageDialog(this, "image size must smaller than 1MB.");

return;

}

filePath = selectedPath;

JTFimg.setText(filePath);

}

}

}

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

// User u = new User();

// u.setUsername("beiyu");

// u.setUsertype(2);

// u.getFromDB();

// new AddCommodityFrame(u);

// }

}

**BussinessFrame.java**

package fattydelivery.bussiness;

import fattydelivery.lib.\*;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\*

\* @author Administrator

\*/

public class BussinessFrame {

private User u;

public BussinessFrame(User u) {

this.u = u;

UIManager.put("Button.font", PJFont.ft);

UIManager.put("Label.font", PJFont.ft);

UIManager.put("TextField.font", PJFont.ft);

UIManager.put("TextArea.font", PJFont.ft);

UIManager.put("ComboBox.font", PJFont.ft);

UIManager.put("PasswordField.font", PJFont.ft);

UIManager.put("RadioButton.font", PJFont.ft);

UIManager.put("TabbedPane.font", PJFont.ft);

JFrame JF = new JFrame("CusteromPage");

JF.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JF.setSize(1600, 900);

JF.setLocationRelativeTo(null);

JF.setResizable(false);

JF.setLayout(null);

Container pane = JF.getContentPane();

//pane.setBackground(new Color(236,229,213));

pane.setBackground(new Color(252, 226, 203));

JTabbedPane JTP = new JTabbedPane();

JTP.addTab("MyCommodityList", new BussinessCommodityList(this.u));

JTP.addTab("OrderList", new BussinessOrderList(this.u));

// JTP.addTab("MyInformation", new CustomerInformationPage(this.u));

JPanel JP = new BussinessInformationPage(this.u);

JTP.setBounds(0, 0, 1200, 900);

JP.setBounds(1225, 50, 360, 800);

JF.add(JTP);

JF.add(JP);

JF.setVisible(true);

JP.setBackground(new Color(252, 226, 203));

}

/\*

\* public static void main(String[] args) { User u = new User();

\* u.setUsername("beiyu"); // u.setUsertype(1); // u.getFromDB(); //

\* u.toString(); u.setPassword("beiyu"); u.setEmail("729320011@qq.com");

\* u.setAddr("Haikou, Hainan"); u.setGender("Male"); u.setRegTime();

\* u.setPhone("1566667912"); u.setUsertype(2); u.setDob("2000-01-06");

\* u.addToDB();

\*

\* Commodity com = new Commodity(); com.setId("food1"); com.setCost(32.66);

\* com.setBussinessId("beiyu"); com.addToDB(); //

\* System.out.println(u.getDob()); //

\* System.out.println(u.getDob().getYear()); BussinessFrame bf = new

\* BussinessFrame(u);

}

\*/

}

**BussinessInformationPage.java**

package fattydelivery.bussiness;

import fattydelivery.lib.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

/\*\*

\*

\* @author Administrator

\*/

public class BussinessInformationPage extends JPanel implements ActionListener {

private User u;

private JLabel JLusername, JLpassword, JLrpassword, JLgender, JLemail, JLphone, JLdob, JLaddr, JLconfirm;

private JTextField JTFusername, JTFemail, JTFphone;

private JPasswordField JPFpassword, JPFrpassword, JPFconfirm;

private JTextArea JTAaddr;

private JComboBox JCBdoby, JCBdobm, JCBdobd;

private JRadioButton JRBmale, JRBfemale;

private ButtonGroup BG1;

private JButton JBsave, JBcancel;

private static int MHEIGHT = 30, MWIDTH = 150, MWIDTH2 = 200, WX = 10, DELTAY = 30;

private String str, regex;

private Pattern pattern;

private Matcher matcher;

private void BuildCom() {

JLusername = new JLabel("Username:");

JLusername.setBounds(WX, 10, MWIDTH, MHEIGHT);

JLpassword = new JLabel("New Password:");

JLpassword.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 1, MWIDTH, MHEIGHT);

JLrpassword = new JLabel("Repeat again:");

JLrpassword.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 2, MWIDTH, MHEIGHT);

JLgender = new JLabel("Gender:");

JLgender.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 3, MWIDTH, MHEIGHT);

JLemail = new JLabel("Email:");

JLemail.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 4, MWIDTH, MHEIGHT);

JLphone = new JLabel("Phone:");

JLphone.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 5, MWIDTH, MHEIGHT);

JLdob = new JLabel("DateofBirth:");

JLdob.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 6, MWIDTH, MHEIGHT);

JLaddr = new JLabel("Address:");

JLaddr.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 8, MWIDTH, MHEIGHT);

JLconfirm = new JLabel("Old Password:");

JLconfirm.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 10, MWIDTH, MHEIGHT);

JTFusername = new JTextField(20);

JTFusername.setBounds(150, 10, MWIDTH2, MHEIGHT);

JTFusername.setEditable(false);

JTFusername.setText(u.getUsername());

JTFemail = new JTextField(20);

JTFemail.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 4, MWIDTH2, MHEIGHT);

JTFemail.setText(u.getEmail());

JTFphone = new JTextField(20);

JTFphone.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 5, MWIDTH2, MHEIGHT);

JTFphone.setText(u.getPhone());

JPFpassword = new JPasswordField(20);

JPFpassword.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 1, MWIDTH2, MHEIGHT);

JPFrpassword = new JPasswordField(20);

JPFrpassword.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 2, MWIDTH2, MHEIGHT);

JPFconfirm = new JPasswordField(20);

JPFconfirm.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 10, MWIDTH2, MHEIGHT);

JTAaddr = new JTextArea(5, 20);

// JTAaddr.setBounds(150, 10+(MHEIGHT+DELTAY)\*7, MWIDTH2, MHEIGHT\*5);

JTAaddr.setText(u.getAddr());

JTAaddr.setLineWrap(true);

JTAaddr.setBorder(BorderFactory.createLineBorder(Color.GRAY));

JScrollPane JSP = new JScrollPane(JTAaddr);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

// JTAaddr.setBounds(150, 10+(MHEIGHT+DELTAY)\*7, MWIDTH2+20, MHEIGHT\*5);

JSP.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 7, MWIDTH2, MHEIGHT \* 5);

JTAaddr.setText(u.getAddr());

// JSP.getHorizontalScrollBar().setBounds(200, 10+(MHEIGHT+DELTAY)\*7, 30, MHEIGHT\*5);

JCBdoby = new JComboBox();

JCBdoby.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 6, 75, MHEIGHT);

// JCBdoby.addItem("----");

for (int i = 1950; i <= 2019; i++) {

JCBdoby.addItem(Integer.toString(i));

}

JCBdoby.setSelectedItem(Integer.toString(u.getDob().getYear() + 1900));

JCBdobm = new JComboBox();

JCBdobm.setBounds(235, 10 + (MHEIGHT + DELTAY) \* 6, 50, MHEIGHT);

for (int i = 1; i <= 12; i++) {

JCBdobm.addItem(Integer.toString(i));

}

JCBdobm.setSelectedItem(Integer.toString(u.getDob().getMonth()));

JCBdobd = new JComboBox();

JCBdobd.setBounds(295, 10 + (MHEIGHT + DELTAY) \* 6, 50, MHEIGHT);

for (int i = 1; i <= 31; i++) {

JCBdobd.addItem(Integer.toString(i));

}

JCBdobd.setSelectedItem(Integer.toString(u.getDob().getDay()));

JRBmale = new JRadioButton("Male");

JRBfemale = new JRadioButton("Female");

BG1 = new ButtonGroup();

BG1.add(JRBmale);

BG1.add(JRBfemale);

if ("male".equalsIgnoreCase(u.getGender())) {

JRBmale.setSelected(true);

} else {

JRBfemale.setSelected(true);

}

JRBmale.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 3, 70, MHEIGHT);

JRBfemale.setBounds(225, 10 + (MHEIGHT + DELTAY) \* 3, 100, MHEIGHT);

JBsave = new JButton("Save");

JBsave.setBounds(20, 10 + (MHEIGHT + DELTAY) \* 11, MWIDTH, 40);

JBsave.addActionListener(this);

JBcancel = new JButton("Cancel");

JBcancel.setBounds(180, 10 + (MHEIGHT + DELTAY) \* 11, MWIDTH, 40);

JBcancel.addActionListener(this);

add(JLusername);

add(JTFusername);

add(JLpassword);

add(JPFpassword);

add(JLrpassword);

add(JPFrpassword);

add(JLgender);

add(JRBmale);

add(JRBfemale);

add(JLemail);

add(JTFemail);

add(JLphone);

add(JTFphone);

add(JLdob);

add(JCBdoby);

add(JCBdobm);

add(JCBdobd);

add(JLaddr); // add(JTAaddr);

add(JLconfirm);

add(JPFconfirm);

add(JBsave);

add(JBcancel);

add(JSP);

this.setBorder(BorderFactory.createLineBorder(Color.GRAY));

// add(JSP.getHorizontalScrollBar());

// JLusername.setForeground(Color.white);

// JLemail.setForeground(Color.white);

// JLphone.setForeground(Color.white);

// JLpassword.setForeground(Color.white);

// JLrpassword.setForeground(Color.white);

// JLconfirm.setForeground(Color.white);

// JLaddr.setForeground(Color.white);

// JLdob.setForeground(Color.white);

// JLgender.setForeground(Color.white);

JTFusername.setBackground(new Color(249, 203, 168));

JTFemail.setBackground(new Color(249, 203, 168));

JTFphone.setBackground(new Color(249, 203, 168));

JPFpassword.setBackground(new Color(249, 203, 168));

JPFconfirm.setBackground(new Color(249, 203, 168));

JTAaddr.setBackground(new Color(249, 203, 168));

JCBdoby.setBackground(new Color(249, 203, 168));

JCBdobm.setBackground(new Color(249, 203, 168));

JPFrpassword.setBackground(new Color(249, 203, 168));

JCBdobd.setBackground(new Color(249, 203, 168));

JRBmale.setBackground(new Color(249, 203, 168));

JRBfemale.setBackground(new Color(249, 203, 168));

JBcancel.setBackground(new Color(241, 156, 67));

JBsave.setBackground(new Color(241, 156, 67));

}

public BussinessInformationPage(User u) {

super(null);

this.u = u;

BuildCom();

}

private void Reset() {

JTFusername.setText(u.getUsername());

JTFemail.setText(u.getEmail());

JTFphone.setText(u.getPhone());

JTAaddr.setText(u.getAddr());

JPFpassword.setText("");

JPFrpassword.setText("");

if ("male".equalsIgnoreCase(u.getGender())) {

JRBmale.setSelected(true);

} else {

JRBfemale.setSelected(true);

}

JCBdoby.setSelectedItem(Integer.toString(u.getDob().getYear() + 1900));

JCBdobm.setSelectedItem(Integer.toString(u.getDob().getMonth() + 1));

JCBdobd.setSelectedItem(Integer.toString(u.getDob().getDay()));

JPFconfirm.setText("");

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBsave) {

u.setPassword(new String(JPFconfirm.getPassword()));

if (JPFconfirm.getPassword().length == 0) {

JOptionPane.showMessageDialog(this, "Please enter your password! ");

return;

}

if (!u.checkPassword()) {

JOptionPane.showMessageDialog(this, "Password is wrong!");

return;

}

if (JPFpassword.getPassword().length != 0 && JPFrpassword.getPassword().length != 0) {

if (new String(JPFpassword.getPassword()).equals(new String(JPFrpassword.getPassword()))) {

u.setPassword(new String(JPFpassword.getPassword()));

} else {

JOptionPane.showMessageDialog(this, "Inconsistent password entered twice!");

return;

}

}

// else {

// JOptionPane.showMessageDialog(this, "Password must not be empty.");

// return;

// }

str = JTFusername.getText();

regex = "^[a-zA-Z0-9\_]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "Username must not be special character.");

return;

}

// str = String.valueOf(JPFpassword.getPassword());

// regex = "^[a-zA-Z0-9]+$";

// pattern = Pattern.compile(regex);

// matcher = pattern.matcher(str);

// if (!matcher.matches()) {

// JOptionPane.showMessageDialog(this, "Password must not be special character.");

// return;

// }

str = JTFemail.getText();

regex = "[a-zA-Z-]{0,}[0-9]{0,}@(([a-zA-Z0-9]-\*){1,}\\.){1,3}[a-zA-Z\\-]{1,}";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "The format of email is incorrect.");

return;

}

str = JTFphone.getText();

regex = "^[0-9]{11}$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "Telephone number must be 11 digits.");

return;

}

if (JRBmale.isSelected()) {

u.setGender("male");

} else {

u.setGender("female");

}

u.setEmail(JTFemail.getText());

u.setPhone(JTFphone.getText());

u.setDob(JCBdoby.getSelectedItem().toString() + "-" + JCBdobm.getSelectedItem().toString() + "-" + JCBdobd.getSelectedItem().toString());

u.setAddr(JTAaddr.getText());

u.updateToDB();

JOptionPane.showMessageDialog(this, "Update success");

Reset();

} else if (e.getSource() == JBcancel) {

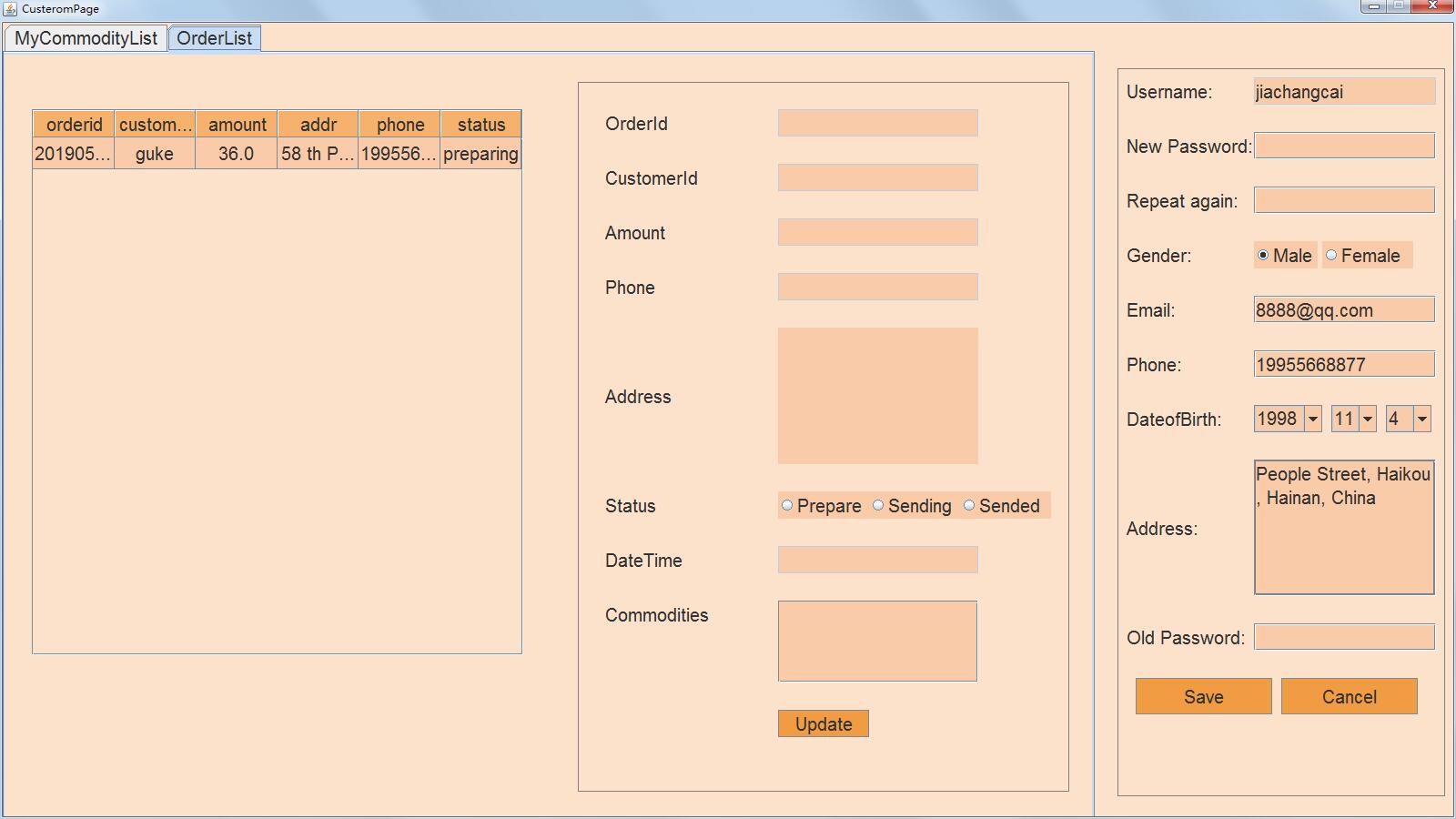
Reset();

}

}

}

**BussinessOrderList.java**

****

package fattydelivery.bussiness;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author Administrator

\*/

public class BussinessOrderList extends JPanel {

private User u;

private Order[] o;

private JTable JTlist;

private JScrollPane JSP;

private JLabel JLcustomerid, JLcommodityid, JLaddr, JLorderid, JLamount, JLdatetime, JLstatus, JLphone;

private JTextField JTFcustomerid, JTForderid, JTFamount, JTFdatetime, JTFphone;

private JTextArea JTAaddr;

private JRadioButton JRBpreparing, JRBdelivering, JRBdelivered;

private ButtonGroup BG;

private JPanel JPlist, JPdetail;

private static String[] ColName = {"orderid", "customerid", "amount", "addr", "phone", "status"};

private String[][] Content;

private DefaultTableModel dtm;

private static int MHEIGHT1 = 30, MWIDTH1 = 30, MWIDTH2 = 150;

private static int MWIDTH3 = 220, MWIDTH4 = 300;

private JTable JTdtl;

private DefaultTableModel dtl;

private JScrollPane JSPdtl;

private JButton JBupdate;

private Order selectOrder;

private void updateTable() {

o = null;

dtm = new DefaultTableModel(null, ColName);

JTlist.setModel(dtm);

o = u.getAllOrder();

if (o.length == 0) {

return;

}

Content = new String[o.length][6];

for (int i = 0; i < o.length; i++) {

Content[i][0] = String.valueOf(o[i].getOrderid());

Content[i][1] = o[i].getCustomId();

Content[i][2] = String.valueOf(o[i].getAmount());

Content[i][3] = o[i].getAddress();

Content[i][4] = o[i].getPhone();

Content[i][5] = o[i].getStatus();

}

dtm = new DefaultTableModel(Content, ColName);

JTlist.setModel(dtm);

}

private void BuildJPlist() {

JPlist = new JPanel(null);

JTlist = new JTable();

updateTable();

DefaultTableCellRenderer r = new DefaultTableCellRenderer();

r.setHorizontalAlignment(JLabel.CENTER);

JTlist.setDefaultRenderer(Object.class, r);

JTlist.getTableHeader().setReorderingAllowed(false);

JTlist.getTableHeader().setResizingAllowed(false);

JTlist.setFont(PJFont.ft);

JTlist.getTableHeader().setFont(PJFont.ft);

JTlist.setRowHeight(35);

JTlist.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

JTlist.getTableHeader().setBounds(30, 30, 540, 30);

// JTlist.setBounds(30, 60, 540, 600);

JSP = new JScrollPane(JTlist);

JSP.setBounds(30, 60, 540, 600);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

// JPlist.add(JTlist);

JPlist.add(JSP);

// JPlist.add(JTlist.getTableHeader());

JPlist.setBounds(0, 0, 600, 900);

this.add(JPlist);

JTlist.setBackground(new Color(249, 203, 168));

JTlist.getTableHeader().setBackground(new Color(246, 177, 108));

JSP.getViewport().setBackground(new Color(252, 226, 203));

setBackground(new Color(252, 226, 203));

JPlist.setBackground(new Color(252, 226, 203));

JTlist.getSelectionModel().addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

int r = JTlist.getSelectedRow();

if (r != -1) {

updateJPdetail(o[r]);

}

}

});

}

private void updateJPdetail(Order o) {

selectOrder = o;

JTForderid.setText(String.valueOf(o.getOrderid()));

JTFcustomerid.setText(o.getCustomId());

JTFamount.setText(String.valueOf(o.getAmount()));

JTAaddr.setText(o.getAddress());

JTFphone.setText(o.getPhone());

JTFdatetime.setText(o.getTime().toString());

if (o.getStatus().equalsIgnoreCase("preparing")) {

JRBpreparing.setSelected(true);

} else if (o.getStatus().equalsIgnoreCase("delivering")) {

JRBdelivering.setSelected(true);

} else {

JRBdelivered.setSelected(true);

}

String[] tmp1 = o.getCommodityIds();

// for(int i=0; i<tmp1.length; i++) System.out.println(tmp1[i]);

String[][] tmp = new String[tmp1.length][1];

for (int i = 0; i < tmp.length; i++) {

tmp[i][0] = tmp1[i];

}

String[] tmp2 = new String[1];

tmp2[0] = "Commodity List";

dtl = new DefaultTableModel(tmp, tmp2);

JTdtl.setModel(dtl);

}

private void BuildJPdetail() {

JPdetail = new JPanel(null);

JLorderid = new JLabel("OrderId");

JLcustomerid = new JLabel("CustomerId");

JLamount = new JLabel("Amount");

JLaddr = new JLabel("Address");

JLphone = new JLabel("Phone");

JLstatus = new JLabel("Status");

JLdatetime = new JLabel("DateTime");

JLcommodityid = new JLabel("Commodities");

JTForderid = new JTextField(30);

JTForderid.setEditable(false);

JTFcustomerid = new JTextField(30);

JTFcustomerid.setEditable(false);

JTFamount = new JTextField(30);

JTFamount.setEditable(false);

JTAaddr = new JTextArea(3, 20);

JTAaddr.setEditable(false);

JTFphone = new JTextField(30);

JTFphone.setEditable(false);

JTFdatetime = new JTextField(30);

JTFdatetime.setEditable(false);

JRBpreparing = new JRadioButton("Prepare");

JRBdelivering = new JRadioButton("Sending");

JRBdelivered = new JRadioButton("Sended");

BG = new ButtonGroup();

BG.add(JRBpreparing);

BG.add(JRBdelivering);

BG.add(JRBdelivered);

JTdtl = new JTable();

DefaultTableCellRenderer r = new DefaultTableCellRenderer();

r.setHorizontalAlignment(JLabel.CENTER);

JTdtl.setDefaultRenderer(Object.class, r);

JTdtl.getTableHeader().setReorderingAllowed(false);

JTdtl.getTableHeader().setResizingAllowed(false);

JTdtl.setFont(PJFont.ft);

JTdtl.getTableHeader().setFont(PJFont.ft);

JTdtl.setRowHeight(35);

JTdtl.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

JSPdtl = new JScrollPane(JTdtl);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

JLorderid.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 1 - 2), MWIDTH2, MHEIGHT1);

JTForderid.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 1 - 2), MWIDTH3, MHEIGHT1);

JLcustomerid.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 2 - 2), MWIDTH2, MHEIGHT1);

JTFcustomerid.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 2 - 2), MWIDTH3, MHEIGHT1);

JLamount.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 3 - 2), MWIDTH2, MHEIGHT1);

JTFamount.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 3 - 2), MWIDTH3, MHEIGHT1);

JLphone.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 4 - 2), MWIDTH2, MHEIGHT1);

JTFphone.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 4 - 2), MWIDTH3, MHEIGHT1);

JLaddr.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 6 - 2), MWIDTH2, MHEIGHT1);

JTAaddr.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 5 - 2), MWIDTH3, MHEIGHT1 \* 5);

JLstatus.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 8 - 2), MWIDTH2, MHEIGHT1);

JRBpreparing.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 8 - 2), 100, MHEIGHT1);

JRBdelivering.setBounds(MWIDTH3 + 100, 30 + MHEIGHT1 \* (2 \* 8 - 2), 100, MHEIGHT1);

JRBdelivered.setBounds(MWIDTH3 + 200, 30 + MHEIGHT1 \* (2 \* 8 - 2), 100, MHEIGHT1);

JLdatetime.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 9 - 2), MWIDTH2, MHEIGHT1);

JTFdatetime.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 9 - 2), MWIDTH3, MHEIGHT1);

JLcommodityid.setBounds(MWIDTH1, 30 + MHEIGHT1 \* (2 \* 10 - 2), MWIDTH2, MHEIGHT1);

JSPdtl.setBounds(MWIDTH3, 30 + MHEIGHT1 \* (2 \* 10 - 2), MWIDTH3, MHEIGHT1 \* 3);

JBupdate = new JButton("Update");

// System.out.println(30+MHEIGHT1\*(2\*12-2));

JBupdate.setBounds(220, 30 + MHEIGHT1 \* (2 \* 12 - 2), 100, 30);

JBupdate.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (selectOrder == null) {

return;

}

if (JRBpreparing.isSelected()) {

selectOrder.setStatus("preparing");

} else if (JRBdelivering.isSelected()) {

selectOrder.setStatus("delivering");

} else if (JRBdelivered.isSelected()) {

selectOrder.setStatus("delivered");

}

selectOrder.updateToDB();

updateTable();

}

});

JPdetail.add(JLorderid);

JPdetail.add(JTForderid);

JPdetail.add(JLcustomerid);

JPdetail.add(JTFcustomerid);

JPdetail.add(JLamount);

JPdetail.add(JTFamount);

JPdetail.add(JLphone);

JPdetail.add(JTFphone);

JPdetail.add(JLaddr);

JPdetail.add(JTAaddr);

JPdetail.add(JLstatus);

JPdetail.add(JRBpreparing);

JPdetail.add(JRBdelivering);

JPdetail.add(JRBdelivered);

JPdetail.add(JLdatetime);

JPdetail.add(JTFdatetime);

JPdetail.add(JLcommodityid);

JPdetail.add(JSPdtl);

JPdetail.add(JBupdate);

JPdetail.setBounds(630, 30, 540, 780);

JPdetail.setBorder(BorderFactory.createLineBorder(Color.GRAY));

this.add(JPdetail);

JPdetail.setBackground(new Color(252, 226, 203));

JTForderid.setBackground(new Color(249, 203, 168));

JTFcustomerid.setBackground(new Color(249, 203, 168));

JTFamount.setBackground(new Color(249, 203, 168));

JTFphone.setBackground(new Color(249, 203, 168));

JTAaddr.setBackground(new Color(249, 203, 168));

JTFdatetime.setBackground(new Color(249, 203, 168));

JSPdtl.getViewport().setBackground(new Color(249, 203, 168));

JRBpreparing.setBackground(new Color(249, 203, 168));

JRBdelivering.setBackground(new Color(249, 203, 168));

JRBdelivered.setBackground(new Color(249, 203, 168));

JBupdate.setBackground(new Color(241, 156, 67));

JTdtl.setBackground(new Color(241, 156, 67));

}

private void Build() {

this.setLayout(null);

BuildJPlist();

BuildJPdetail();

}

public BussinessOrderList(User u) {

this.u = u;

Build();

}

/\*

\* public static void main(String args[]) { JFrame JF = new JFrame("qwq");

\* User u = new User(); u.setUsername("beiyu"); u.setPassword("beiyu");

\* u.setUsertype(2); u.getFromDB(); System.out.println(u.toString());

\* JF.add(new BussinessOrderList(u)); JF.setSize(1600, 900);

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

\* JF.setLocationRelativeTo(null); JF.setVisible(true);

}

\*/

}

**fattydelivery.customer**

****

**CommodityList.java**

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

/\*\*

\*

\* @author Administrator

\*/

public class CommodityList extends JScrollPane {

private CommoditySelectPanel FJP;

private Commodity[] oo;

private JPanel JP;

public CommodityPanel cp[];

private void Build() {

JP = new JPanel(null);

oo = User.getAllCommodities();

if(oo.length == 0);

cp = new CommodityPanel[oo.length];

for(int i=0; i<oo.length; i++) {

cp[i] = new CommodityPanel(oo[i], FJP);

cp[i].setBounds(0, i\*200, cp[i].getWidth(), cp[i].getHeight());

JP.add(cp[i]);

JP.updateUI();

JP.validate();

this.setViewportView(JP);

JP.setPreferredSize(new Dimension(JP.getWidth(), JP.getHeight() + cp[i].getHeight()));

System.out.println(oo[i].toString());

}

JP.setPreferredSize(new Dimension(400,oo.length\*200));

JP.setBackground(new Color(252,226,203));

// JSP.setViewportView(JP);

// JSP.getViewport().setLayout(new BorderLayout());

// JSP.getViewport().add(JP, BorderLayout.CENTER);

// this.setSize(400, 800);

this.setBounds(50, 50, 400, 600);

this.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

this.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

this.getVerticalScrollBar().setUnitIncrement(15);

}

public CommodityList(CommoditySelectPanel FJP) {

this.FJP = FJP;

Build();

}

}

**CommodityPanel.java**

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.\*;

/\*\*

\*

\* @author Administrator

\*/

public class CommodityPanel extends JPanel implements ActionListener {

private CommoditySelectPanel FJP;

private JLabel JLcommodityid, JLcommodityname, JLprice, JLpricea, JLimg;

private String filePath;

public JButton JBadd, JBdel;

private JLabel JLnum;

private int num;

public CommodityPanel(Commodity c, CommoditySelectPanel FJP) {

this.FJP = FJP;

this.setLayout(null);

JLcommodityid = new JLabel("Name");

JLcommodityname = new JLabel(c.getId());

JLprice = new JLabel("Price");

JLpricea = new JLabel(String.valueOf(c.getCost()));

JBadd = new JButton("+");

JBdel = new JButton("-");

JLnum = new JLabel(String.valueOf(num));

File ff = new File("img\\" + c.getBussinessId());

if (!ff.exists()) {

ff.mkdirs();

}

// filePath = "img\\" + c.getId() + "." + c.getImgext();

filePath = "img\\" + c.getBussinessId() + "\\" + c.getId() + "." + c.getImgext();

// System.out.println(filePath);

c.writeToLocal(filePath);

ImageIcon imageIcon = new ImageIcon(filePath);

Image image = imageIcon.getImage();

Image scaledInstance = image.getScaledInstance(100, 100, Image.SCALE\_AREA\_AVERAGING);

imageIcon.setImage(scaledInstance);

JLimg = new JLabel(imageIcon);

JLcommodityid.setBounds(150, 30, 100, 30);

JLcommodityname.setBounds(250, 30, 100, 30);

JLprice.setBounds(150, 80, 100, 30);

JLpricea.setBounds(250, 80, 100, 30);

JLimg.setBounds(25, 30, 100, 100);

JBadd.setBounds(315, 110, 50, 30);

JLnum.setBounds(280, 110, 40, 30);

JBdel.setBounds(215, 110, 45, 30);

JBadd.addActionListener(FJP);

JBadd.addActionListener(this);

JBdel.addActionListener(this);

JBadd.setActionCommand(c.getId());

this.add(JLcommodityid);

this.add(JLcommodityname);

this.add(JLprice);

this.add(JLpricea);

this.add(JLimg);

this.add(JBadd);

this.add(JBdel);

this.add(JLnum);

this.setSize(400, 150);

this.setBorder(BorderFactory.createLineBorder(Color.GRAY));

JBadd.setBackground(new Color(241, 156, 67));

JBdel.setBackground(new Color(241, 156, 67));

this.setBackground(new Color(252, 226, 203));

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBadd) {

if (num >= 99) {

JOptionPane.showMessageDialog(this, "This is too large...");

return;

}

num += 1;

JLnum.setText(String.valueOf(num));

} else if (e.getSource() == JBdel) {

if (num <= 0) {

return;

}

num -= 1;

JLnum.setText(String.valueOf(num));

}

}

public int getNum() {

return num;

}

public String getName() {

return JLcommodityname.getText();

}

}

**CommoditySelectPanel.java**

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Vector;

import javax.swing.\*;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author Administrator

\*/

public class CommoditySelectPanel extends JPanel implements ActionListener {

private User u;

private CommodityList cp;

private JScrollPane JSP;

private JTable JTlist;

private static String[] ColName = {"id", "cost"};

private String[][] Content;

private Vector<Commodity> oo;

private JButton JBpay, JBadd, JBclr;

private DefaultTableModel dtm;

private JLabel JLtot;

private double cost;

private void updateTable() {

dtm = new DefaultTableModel(null, ColName);

JTlist.setModel(dtm);

if (oo.size() == 0) {

cost = 0;

JLtot.setText("Total: 0");

return;

}

Content = new String[oo.size()][2];

cost = 0;

for (int i = 0; i < oo.size(); i++) {

// System.out.println(oo.get(i).toString());

Content[i][0] = oo.get(i).getId();

Content[i][1] = String.valueOf(oo.get(i).getCost());

cost += oo.get(i).getCost();

}

dtm = new DefaultTableModel(Content, ColName);

JTlist.setModel(dtm);

JLtot.setText("Total: " + String.valueOf(cost));

}

public CommoditySelectPanel(User u) {

oo = new Vector<Commodity>();

cp = new CommodityList(this);

cost = 0;

this.u = u;

this.add(cp);

this.setLayout(null);

this.setBounds(0, 0, cp.getWidth() + 100, cp.getHeight() + 300);

JLtot = new JLabel("Total: " + String.valueOf(cost));

JBpay = new JButton("Buy Now");

JBadd = new JButton("Add");

JBclr = new JButton("clear");

JLtot.setBounds(100, 800, 200, 30);

JBpay.setBounds(300, 800, 150, 30);

JBadd.setBounds(100, 650, 100, 30);

JBclr.setBounds(300, 650, 100, 30);

JBpay.addActionListener(this);

JBclr.addActionListener(this);

JBadd.addActionListener(this);

JTlist = new JTable();

updateTable();

DefaultTableCellRenderer r = new DefaultTableCellRenderer();

r.setHorizontalAlignment(JLabel.CENTER);

JTlist.setDefaultRenderer(Object.class, r);

JTlist.getTableHeader().setReorderingAllowed(false);

JTlist.getTableHeader().setResizingAllowed(false);

// JTlist.setFont(PJFont.ft);

JTlist.getTableHeader().setFont(PJFont.ft);

JTlist.setRowHeight(20);

JTlist.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

// JTlist.getTableHeader().setBounds(30, 30, 540, 30);

// JTlist.setBounds(30, 60, 540, 600);

JSP = new JScrollPane(JTlist);

JSP.setBounds(50, 680, 400, 100);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

// JPlist.add(JTlist);

this.add(JSP);

this.add(JLtot);

this.add(JBpay);

this.add(JBclr);

this.add(JBadd);

JTlist.setBackground(new Color(252, 226, 203));

JSP.getViewport().setBackground(new Color(252, 226, 203));

JBpay.setBackground(new Color(241, 156, 67));

this.setBackground(new Color(252, 226, 203));

JTlist.setBackground(new Color(249, 203, 168));

JTlist.getTableHeader().setBackground(new Color(246, 177, 108));

}

private void addOrder(String bsid, Vector<String> com, double totcost) {

Order ord = new Order();

ord.setAddress(u.getAddr());

ord.setAmount(totcost);

ord.setBussinessId(bsid);

ord.setCustomId(u.getUsername());

String[] str = new String[com.size()];

for (int i = 0; i < com.size(); i++) {

str[i] = com.get(i);

}

ord.setStatus("preparing");

ord.setCommodityId(str);

ord.setPhone(u.getPhone());

ord.addToDB();

}

private void addOrders() {

Vector<String> ids = new Vector<String>();

Vector<String> tmp = null;

for (int i = 0; i < oo.size(); i++) {

Commodity tt = oo.get(i);

String bsi = tt.getBussinessId();

boolean f = true;

for (int j = 0; j < ids.size(); j++) {

if (bsi.equals(ids.get(j))) {

f = false;

break;

}

}

if (f) {

ids.add(bsi);

}

}

// System.out.println("-----------");

// for(int i=0; i<ids.size(); i++) System.out.println(ids.get(i));

for (int i = 0; i < ids.size(); i++) {

tmp = new Vector<String>();

double totcost = 0;

for (int j = 0; j < oo.size(); j++) {

if (oo.get(j).getBussinessId().equals(ids.get(i))) {

tmp.add(oo.get(j).getId());

totcost += oo.get(i).getCost();

// System.out.println("???????");

}

}

addOrder(ids.get(i), tmp, totcost);

}

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBpay) {

if (oo.size() == 0) {

JOptionPane.showMessageDialog(this, "You haven't selected anything!");

return;

}

addOrders();

JOptionPane.showMessageDialog(this, "Add Order Success!");

oo.clear();

this.updateTable();

return;

} else if (e.getSource() == JBclr) {

oo.clear();

this.updateTable();

// System.out.println(e.getActionCommand());

// Commodity c = new Commodity();

// c.setId(e.getActionCommand());

// c.getFromDB();

// oo.add(c);

// this.updateTable();

} else if (e.getSource() == JBadd) {

oo.clear();

for (int i = 0; i < cp.cp.length; i++) {

Commodity c = new Commodity();

c.setId(cp.cp[i].getName());

c.getFromDB();

for (int j = 0; j < cp.cp[i].getNum(); j++) {

oo.add(c);

}

}

this.updateTable();

}

}

//

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

// JFrame JF = new JFrame("qwq");

// User u = new User();

// u.setUsername("beiyu");

// u.setPassword("beiyu");

// u.setUsertype(1);

// u.getFromDB();

// System.out.println(u.toString());

// CommoditySelectPanel csp = new CommoditySelectPanel(u);

// JF.add(csp);

// JF.setSize(500, 900);

// JF.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// JF.setLocationRelativeTo(null);

// JF.setLayout(null);

// JF.setVisible(true);

// }

}

**CustomerFrame.java**

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.Container;

import javax.swing.\*;

/\*\*

\*

\* @author Administrator

\*/

public class CustomerFrame extends JFrame {

private User u;

private JTabbedPane JTP;

public CustomerFrame(User u) {

super("Welcome!");

this.u = u;

this.setSize(500, 900);

this.setLocationRelativeTo(null);

this.setLayout(null);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setResizable(false);

JTP = new JTabbedPane();

JTP.add("CommodityList", new CommoditySelectPanel(this.u));

JTP.add("Order", new CustomerOrderList(u));

CustomerInformationPage CIP = new CustomerInformationPage(u);

CIP.setBounds(50, 25, 400, 800);

JPanel JP = new JPanel(null);

JP.add(CIP);

JTP.add("User Infor", JP);

JTP.setBounds(0, 0, 500, 900);

this.add(JTP);

this.setVisible(true);

Container pane = this.getContentPane();

pane.setBackground(new Color(252, 226, 203));

JP.setBackground(new Color(252, 226, 203));

}

/\*

\* public static void main(String[] args) { User u = new User();

\* u.setUsername("beiyu"); u.setPassword("beiyu"); u.setUsertype(1);

\* u.getFromDB(); System.out.println(u.toString()); new CustomerFrame(u); }

\*/

}

**CustomerInformationPage.java**

****

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

/\*\*

\*

\* @author Administrator

\*/

public class CustomerInformationPage extends JPanel implements ActionListener {

private User u;

private JLabel JLusername, JLpassword, JLrpassword, JLgender, JLemail, JLphone, JLdob, JLaddr, JLconfirm;

private JTextField JTFusername, JTFemail, JTFphone;

private JPasswordField JPFpassword, JPFrpassword, JPFconfirm;

private JTextArea JTAaddr;

private JComboBox JCBdoby, JCBdobm, JCBdobd;

private JRadioButton JRBmale, JRBfemale;

private ButtonGroup BG1;

private JButton JBsave, JBcancel;

private static int MHEIGHT = 30, MWIDTH = 150, MWIDTH2 = 200, WX = 10, DELTAY = 30;

String str, regex;

Pattern pattern;

Matcher matcher;

private void BuildCom() {

JLusername = new JLabel("Username:");

JLusername.setBounds(WX, 10, MWIDTH, MHEIGHT);

JLpassword = new JLabel("New Password:");

JLpassword.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 1, MWIDTH, MHEIGHT);

JLrpassword = new JLabel("Repeat again:");

JLrpassword.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 2, MWIDTH, MHEIGHT);

JLgender = new JLabel("Gender:");

JLgender.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 3, MWIDTH, MHEIGHT);

JLemail = new JLabel("Email:");

JLemail.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 4, MWIDTH, MHEIGHT);

JLphone = new JLabel("Phone:");

JLphone.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 5, MWIDTH, MHEIGHT);

JLdob = new JLabel("DateofBirth:");

JLdob.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 6, MWIDTH, MHEIGHT);

JLaddr = new JLabel("Address:");

JLaddr.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 8, MWIDTH, MHEIGHT);

JLconfirm = new JLabel("Old Password:");

JLconfirm.setBounds(WX, 10 + (MHEIGHT + DELTAY) \* 10, MWIDTH, MHEIGHT);

JTFusername = new JTextField(20);

JTFusername.setBounds(150, 10, MWIDTH2, MHEIGHT);

JTFusername.setEditable(false);

JTFusername.setText(u.getUsername());

JTFemail = new JTextField(20);

JTFemail.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 4, MWIDTH2, MHEIGHT);

JTFemail.setText(u.getEmail());

JTFphone = new JTextField(20);

JTFphone.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 5, MWIDTH2, MHEIGHT);

JTFphone.setText(u.getPhone());

JPFpassword = new JPasswordField(20);

JPFpassword.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 1, MWIDTH2, MHEIGHT);

JPFrpassword = new JPasswordField(20);

JPFrpassword.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 2, MWIDTH2, MHEIGHT);

JPFconfirm = new JPasswordField(20);

JPFconfirm.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 10, MWIDTH2, MHEIGHT);

JTAaddr = new JTextArea(5, 20);

// JTAaddr.setBounds(150, 10+(MHEIGHT+DELTAY)\*7, MWIDTH2, MHEIGHT\*5);

JTAaddr.setText(u.getAddr());

JTAaddr.setLineWrap(true);

JTAaddr.setBorder(BorderFactory.createLineBorder(Color.GRAY));

JScrollPane JSP = new JScrollPane(JTAaddr);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

// JTAaddr.setBounds(150, 10+(MHEIGHT+DELTAY)\*7, MWIDTH2+20, MHEIGHT\*5);

JSP.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 7, MWIDTH2, MHEIGHT \* 5);

JTAaddr.setText(u.getAddr());

// JSP.getHorizontalScrollBar().setBounds(200, 10+(MHEIGHT+DELTAY)\*7, 30, MHEIGHT\*5);

JCBdoby = new JComboBox();

JCBdoby.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 6, 75, MHEIGHT);

// JCBdoby.addItem("----");

for (int i = 1950; i <= 2019; i++) {

JCBdoby.addItem(Integer.toString(i));

}

JCBdoby.setSelectedItem(Integer.toString(u.getDob().getYear() + 1900));

JCBdobm = new JComboBox();

JCBdobm.setBounds(235, 10 + (MHEIGHT + DELTAY) \* 6, 50, MHEIGHT);

for (int i = 1; i <= 12; i++) {

JCBdobm.addItem(Integer.toString(i));

}

JCBdobm.setSelectedItem(Integer.toString(u.getDob().getMonth()));

JCBdobd = new JComboBox();

JCBdobd.setBounds(295, 10 + (MHEIGHT + DELTAY) \* 6, 50, MHEIGHT);

for (int i = 1; i <= 31; i++) {

JCBdobd.addItem(Integer.toString(i));

}

JCBdobd.setSelectedItem(Integer.toString(u.getDob().getDay()));

JRBmale = new JRadioButton("Male");

JRBfemale = new JRadioButton("Female");

BG1 = new ButtonGroup();

BG1.add(JRBmale);

BG1.add(JRBfemale);

if ("male".equalsIgnoreCase(u.getGender())) {

JRBmale.setSelected(true);

} else {

JRBfemale.setSelected(true);

}

JRBmale.setBounds(150, 10 + (MHEIGHT + DELTAY) \* 3, 70, MHEIGHT);

JRBfemale.setBounds(225, 10 + (MHEIGHT + DELTAY) \* 3, 100, MHEIGHT);

JBsave = new JButton("Save");

JBsave.setBounds(20, 10 + (MHEIGHT + DELTAY) \* 11, MWIDTH, 40);

JBsave.addActionListener(this);

JBcancel = new JButton("Cancel");

JBcancel.setBounds(180, 10 + (MHEIGHT + DELTAY) \* 11, MWIDTH, 40);

JBcancel.addActionListener(this);

add(JLusername);

add(JTFusername);

add(JLpassword);

add(JPFpassword);

add(JLrpassword);

add(JPFrpassword);

add(JLgender);

add(JRBmale);

add(JRBfemale);

add(JLemail);

add(JTFemail);

add(JLphone);

add(JTFphone);

add(JLdob);

add(JCBdoby);

add(JCBdobm);

add(JCBdobd);

add(JLaddr); // add(JTAaddr);

add(JLconfirm);

add(JPFconfirm);

add(JBsave);

add(JBcancel);

add(JSP);

this.setBorder(BorderFactory.createLineBorder(Color.GRAY));

// add(JSP.getHorizontalScrollBar());

JTFusername.setBackground(new Color(249, 203, 168));

JTFemail.setBackground(new Color(249, 203, 168));

JTFphone.setBackground(new Color(249, 203, 168));

JPFpassword.setBackground(new Color(249, 203, 168));

JPFconfirm.setBackground(new Color(249, 203, 168));

JTAaddr.setBackground(new Color(249, 203, 168));

JCBdoby.setBackground(new Color(249, 203, 168));

JCBdobm.setBackground(new Color(249, 203, 168));

JPFrpassword.setBackground(new Color(249, 203, 168));

JCBdobd.setBackground(new Color(249, 203, 168));

JRBmale.setBackground(new Color(249, 203, 168));

JRBfemale.setBackground(new Color(249, 203, 168));

JBcancel.setBackground(new Color(241, 156, 67));

JBsave.setBackground(new Color(241, 156, 67));

setBackground(new Color(252, 226, 203));

}

public CustomerInformationPage(User u) {

super(null);

this.u = u;

BuildCom();

}

private void Reset() {

JTFusername.setText(u.getUsername());

JTFemail.setText(u.getEmail());

JTFphone.setText(u.getPhone());

JTAaddr.setText(u.getAddr());

JPFpassword.setText("");

JPFrpassword.setText("");

if ("male".equalsIgnoreCase(u.getGender())) {

JRBmale.setSelected(true);

} else {

JRBfemale.setSelected(true);

}

JCBdoby.setSelectedItem(Integer.toString(u.getDob().getYear() + 1900));

JCBdobm.setSelectedItem(Integer.toString(u.getDob().getMonth() + 1));

JCBdobd.setSelectedItem(Integer.toString(u.getDob().getDay()));

JPFconfirm.setText("");

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBsave) {

u.setPassword(new String(JPFconfirm.getPassword()));

if (JPFconfirm.getPassword().length == 0) {

JOptionPane.showMessageDialog(this, "Please enter your password! ");

return;

}

if (!u.checkPassword()) {

JOptionPane.showMessageDialog(this, "Password is wrong!");

return;

}

if (JPFpassword.getPassword().length != 0 && JPFrpassword.getPassword().length != 0) {

if (new String(JPFpassword.getPassword()).equals(new String(JPFrpassword.getPassword()))) {

u.setPassword(new String(JPFpassword.getPassword()));

} else {

JOptionPane.showMessageDialog(this, "Inconsistent password entered twice!");

return;

}

}

// else {

// JOptionPane.showMessageDialog(this, "Password must not be empty.");

// return;

// }

// str = JTFusername.getText();

// regex = "^[a-zA-Z0-9\_]+$";

// pattern = Pattern.compile(regex);

// matcher = pattern.matcher(str);

// if (!matcher.matches()) {

// JOptionPane.showMessageDialog(this, "Username must not be special character.");

// return;

// }

// str = String.valueOf(JPFpassword.getPassword());

// regex = "^[a-zA-Z0-9]+$";

// pattern = Pattern.compile(regex);

// matcher = pattern.matcher(str);

// if (!matcher.matches()) {

// JOptionPane.showMessageDialog(this, "Password must not be special character.");

// return;

// }

str = JTFemail.getText();

regex = "[a-zA-Z-]{0,}[0-9]{0,}@(([a-zA-Z0-9]-\*){1,}\\.){1,3}[a-zA-Z\\-]{1,}";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "The format of email is incorrect.");

return;

}

str = JTFphone.getText();

regex = "^[0-9]{11}$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(this, "Telephone number must be 11 digits.");

return;

}

if (JRBmale.isSelected()) {

u.setGender("male");

} else {

u.setGender("female");

}

u.setEmail(JTFemail.getText());

u.setPhone(JTFphone.getText());

u.setDob(JCBdoby.getSelectedItem().toString() + "-" + JCBdobm.getSelectedItem().toString() + "-" + JCBdobd.getSelectedItem().toString());

u.setAddr(JTAaddr.getText());

u.updateToDB();

JOptionPane.showMessageDialog(this, "Update success");

Reset();

} else if (e.getSource() == JBcancel) {

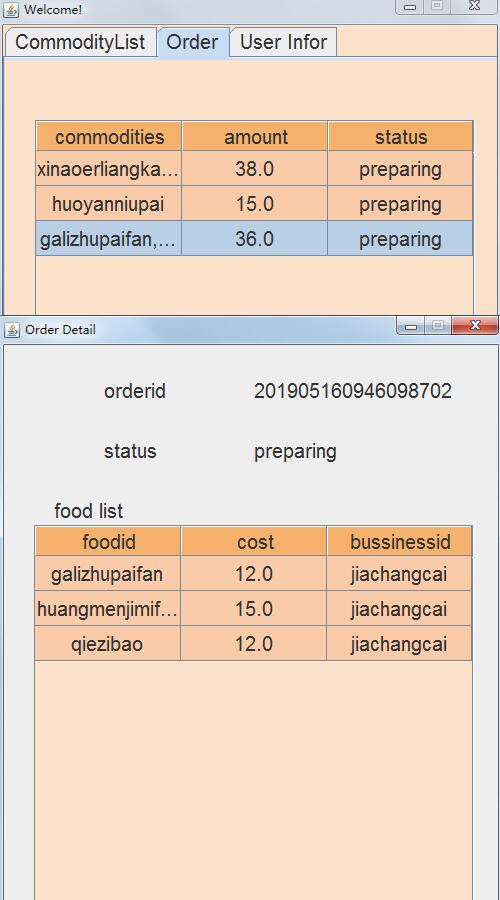
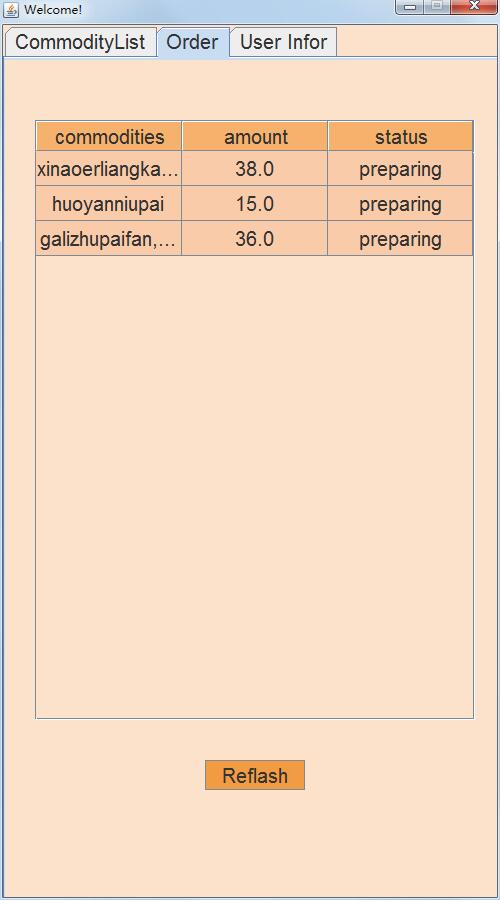
Reset();

}

}

}

**CustomerOrderList.java**

****

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author Administrator

\*/

public class CustomerOrderList extends JPanel {

private User u;

private Order[] o;

private JTable JTlist;

private JScrollPane JSP;

private static String[] ColName = {"commodities", "amount", "status"};

private String[][] Content;

private DefaultTableModel dtm;

private JButton JBreflash;

private void updateTable() {

o = null;

dtm = new DefaultTableModel(null, ColName);

JTlist.setModel(dtm);

o = u.getAllOrder();

if (o.length == 0) {

return;

}

Content = new String[o.length][3];

for (int i = 0; i < o.length; i++) {

Content[i][0] = o[i].getCommodityId();

Content[i][1] = String.valueOf(o[i].getAmount());

Content[i][2] = o[i].getStatus();

}

dtm = new DefaultTableModel(Content, ColName);

JTlist.setModel(dtm);

}

private void Build() {

this.setLayout(null);

JTlist = new JTable();

updateTable();

DefaultTableCellRenderer r = new DefaultTableCellRenderer();

r.setHorizontalAlignment(JLabel.CENTER);

JTlist.setDefaultRenderer(Object.class, r);

JTlist.getTableHeader().setReorderingAllowed(false);

JTlist.getTableHeader().setResizingAllowed(false);

JTlist.setFont(PJFont.ft);

JTlist.getTableHeader().setFont(PJFont.ft);

JTlist.setRowHeight(35);

JTlist.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

JTlist.getTableHeader().setBounds(30, 30, 440, 30);

// JTlist.setBounds(30, 60, 540, 600);

JSP = new JScrollPane(JTlist);

JSP.setBounds(30, 60, 440, 600);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

// JPlist.add(JTlist);

add(JSP);

// JPlist.add(JTlist.getTableHeader());

JBreflash = new JButton("Reflash");

JBreflash.setBounds(200, 700, 100, 30);

JBreflash.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

updateTable();

}

});

this.add(JBreflash);

setBounds(0, 0, 500, 900);

JTlist.getSelectionModel().addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

if (e.getValueIsAdjusting() == false) {

int r = JTlist.getSelectedRow();

if (r != -1) {

// System.out.println(o[r].toString());

new OrderDetailFrame(o[r]);

}

}

}

});

JTlist.setBackground(new Color(249,203,168));

JTlist.getTableHeader().setBackground(new Color(246,177,108));

JSP.getViewport().setBackground(new Color(252,226,203));

setBackground(new Color(252,226,203));

//JPlist.setBackground(new Color(252,226,203));

JBreflash.setBackground(new Color(241,156,67));

}

public CustomerOrderList(User u) {

this.u = u;

Build();

}

/\*

public static void main(String args[]) {

JFrame JF = new JFrame("qwq");

User u = new User();

u.setUsername("beiyu");

u.setPassword("beiyu");

u.setUsertype(2);

u.getFromDB();

System.out.println(u.toString());

JF.add(new CustomerOrderList(u));

JF.setSize(800, 900);

JF.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JF.setLocationRelativeTo(null);

JF.setVisible(true);

}\*/

}

**OrderDetailFrame.java**

package fattydelivery.customer;

import fattydelivery.lib.\*;

import java.awt.Color;

import java.awt.HeadlessException;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author Administrator

\*/

public class OrderDetailFrame extends JFrame implements ActionListener {

private Order ord;

private JPanel JP;

private JLabel JLorderid, JLfoodlist, JLorderid1, JLstatus, JLstatus1;

private JTable JTlist;

private JScrollPane JSP;

private String[] ColName = {"foodid", "cost", "bussinessid"};

private String[][] Content;

private String[] com;

private DefaultTableModel dtm;

private JButton JBcancel, JBclose;

void updateTable() {

dtm = new DefaultTableModel(null, ColName);

if (com.length == 0) {

return;

}

Content = new String[com.length][3];

for (int i = 0; i < com.length; i++) {

Commodity tmp = new Commodity();

tmp.setId(com[i].trim());

tmp.getFromDB();

Content[i][0] = tmp.getId();

Content[i][1] = String.valueOf(tmp.getCost());

Content[i][2] = tmp.getBussinessId();

}

dtm = new DefaultTableModel(Content, ColName);

JTlist.setModel(dtm);

}

private void Build() {

com = ord.getCommodityIds();

JP = new JPanel(null);

// for(int i=0; i<com.length; i++) System.out.println(com[i]);

JLorderid = new JLabel("orderid");

JLorderid1 = new JLabel(String.valueOf(ord.getOrderid()));

JLstatus = new JLabel("status");

JLstatus1 = new JLabel(ord.getStatus());

JLfoodlist = new JLabel("food list");

JLorderid.setBounds(100, 30, 120, 30);

JLorderid1.setBounds(250, 30, 250, 30);

JLstatus.setBounds(100, 90, 120, 30);

JLstatus1.setBounds(250, 90, 250, 30);

JLfoodlist.setBounds(50, 150, 120, 30);

JTlist = new JTable();

updateTable();

DefaultTableCellRenderer r = new DefaultTableCellRenderer();

r.setHorizontalAlignment(JLabel.CENTER);

JTlist.setDefaultRenderer(Object.class, r);

JTlist.getTableHeader().setReorderingAllowed(false);

JTlist.getTableHeader().setResizingAllowed(false);

JTlist.setFont(PJFont.ft);

JTlist.getTableHeader().setFont(PJFont.ft);

JTlist.setRowHeight(35);

JTlist.setAutoResizeMode(JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

// JTlist.getTableHeader().setBounds(30, 30, 540, 30);

// JTlist.setBounds(30, 60, 540, 600);

JSP = new JScrollPane(JTlist);

JSP.setBounds(30, 180, 440, 520);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

JBcancel = new JButton("Cancel");

JBclose = new JButton("Close");

JBcancel.setBounds(50, 730, 120, 30);

JBclose.setBounds(300, 730, 120, 30);

JBcancel.addActionListener(this);

JBclose.addActionListener(this);

if (!ord.getStatus().equalsIgnoreCase("preparing")) {

JBcancel.setEnabled(false);

}

JP.add(JLorderid);

JP.add(JLorderid1);

JP.add(JLstatus);

JP.add(JLstatus1);

JP.add(JLfoodlist);

JP.add(JSP);

JP.add(JBcancel);

JP.add(JBclose);

this.add(JP);

JTlist.setBackground(new Color(249, 203, 168));

JTlist.getTableHeader().setBackground(new Color(246, 177, 108));

setBackground(new Color(252, 226, 203));

JBcancel.setBackground(new Color(241, 156, 67));

JBclose.setBackground(new Color(241, 156, 67));

JSP.getViewport().setBackground(new Color(252, 226, 203));

}

public OrderDetailFrame(Order ord) {

super("Order Detail");

// System.out.println(ord.toString());

this.ord = ord;

this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

this.setLocationRelativeTo(null);

this.setSize(500, 800);

this.setResizable(false);

Build();

this.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == JBclose) {

this.dispose();

} else {

ord.deleteFromDB();

this.dispose();

}

}

}

**fattydelivery.lib**

**Commodity.java**

package fattydelivery.lib;

import java.io.\*;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Administrator

\*/

public class Commodity {

@Override

public String toString() {

return "Commodity{" + "id=" + id + ", cost=" + cost + ", bussinessid=" + bussinessid + '}';

}

private String id;

private double cost;

private String bussinessid;

private File imgf;

private InputStream imgs;

private String imgext;

public void setBussinessId(String a) {

bussinessid = a;

}

public String getBussinessId() {

return bussinessid;

}

public void setId(String a) {

id = a;

}

public String getId() {

return id;

}

public void setCost(double a) {

cost = a;

}

public double getCost() {

return cost;

}

public void setImg(InputStream imgs) {

this.imgs = imgs;

}

public void setImg(String filepath) {

try {

imgf = new File(filepath);

imgs = new FileInputStream(imgf);

imgext = filepath.substring(filepath.lastIndexOf('.') + 1);

System.out.println(imgext);

} catch (FileNotFoundException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

}

}

public InputStream getImg() {

return imgs;

}

public void setImgext(String Imgext) {

this.imgext = Imgext;

}

public String getImgext() {

return imgext;

}

public void writeToLocal(String path) {

FileOutputStream fos = null;

// System.out.println(path);

try {

// System.out.println(path);

fos = new FileOutputStream(path);

byte[] b = new byte[1024];

int length;

while ((length = imgs.read(b)) > 0) {

fos.write(b, 0, length);

}

imgs.close();

} catch (FileNotFoundException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

fos.close();

} catch (IOException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

private boolean check(Connection con) {

boolean f = false;

PreparedStatement stm = null;

ResultSet rs = null;

try {

String sql;

sql = "select \* from CommodityTable where id=?";

stm = con.prepareStatement(sql);

stm.setString(1, id);

rs = stm.executeQuery();

if (rs.next()) {

f = false;

} else {

f = true;

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

return f;

}

public int addToDB() {

try {

Connection con;

con = DBConnect.connect();

if (!check(con)) {

return 0;

}

String sql = "insert into CommodityTable values(?, ?, ?, ?, ?)";

PreparedStatement stm = con.prepareStatement(sql);

stm.setString(1, id);

stm.setDouble(2, cost);

stm.setString(3, bussinessid);

stm.setBinaryStream(4, imgs);

stm.setString(5, imgext);

stm.executeUpdate();

DBConnect.disconnect(con, stm);

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

return 1;

}

public void updateToDB(Connection con) {

}

public void deleteFromDB() {

Connection con = DBConnect.connect();

String sql = null;

PreparedStatement stm = null;

try {

sql = "delete from CommodityTable where id = ?";

stm = con.prepareStatement(sql);

stm.setString(1, id);

stm.executeUpdate();

} catch (SQLException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public void getFromDB() {

Connection con = DBConnect.connect();

String sql = null;

PreparedStatement stm = null;

ResultSet rs = null;

try {

sql = "select \* from CommodityTable where id = ?";

stm = con.prepareStatement(sql);

stm.setString(1, id);

rs = stm.executeQuery();

for (; rs.next();) {

this.setCost(rs.getDouble(2));

this.setBussinessId(rs.getString(3));

this.setImg(rs.getBinaryStream(4));

this.setImgext(rs.getString(5));

}

} catch (SQLException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(Commodity.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

**DBConnect.java**

package fattydelivery.lib;

import java.sql.\*;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Administrator

\*/

public class DBConnect {

public static Connection connect() {

String username = "root";

String password = "1234";

String url = "jdbc:mysql://localhost:3306/fattydelivery";

Connection con = null;

try {

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection(url, username, password);

if (con != null) {

System.out.println("Connect to MySQL successfully");

} else {

System.out.println("Connect to MySQL failed");

}

} catch (ClassNotFoundException ex) {

Logger.getLogger(DBConnect.class.getName()).log(Level.SEVERE, null, ex);

} catch (SQLException ex) {

Logger.getLogger(DBConnect.class.getName()).log(Level.SEVERE, null, ex);

} finally {

return con;

}

}

public static void disconnect(Connection con) {

if (con != null) {

try {

con.close();

} catch (SQLException ex) {

Logger.getLogger(DBConnect.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static void disconnect(Connection con, PreparedStatement stm) {

if (stm != null) {

try {

stm.close();

} catch (SQLException ex) {

Logger.getLogger(DBConnect.class.getName()).log(Level.SEVERE, null, ex);

}

}

if (con != null) {

try {

con.close();

} catch (SQLException ex) {

Logger.getLogger(DBConnect.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

Order.java

package fattydelivery.lib;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Administrator

\*/

public class Order {

@Override

public String toString() {

return "Order{" + "addr=" + addr + ", amount=" + amount + ", bussinessid=" + bussinessid + ", customerid=" + customerid + ", commodityid=" + commodityid + ", date=" + date + ", status=" + status + ", time=" + time + ", phone=" + phone + ", orderid=" + orderid + '}';

}

private String addr;

private double amount;

private String bussinessid;

private String customerid;

private String commodityid;

private java.util.Date date;

private String status;

private Timestamp time;

private String phone;

private long orderid;

public long getOrderid() {

return orderid;

}

public void setOrderid(long orderid) {

this.orderid = orderid;

}

public void setOrderid() {

setTime();

SimpleDateFormat f = new SimpleDateFormat("yyyyMMddHHmmss");

Random r = new Random();

int x = r.nextInt(9999);

String str = String.format("%04d", x);

orderid = Long.parseLong(f.format(time) + str);

}

public Order() {

setTime();

SimpleDateFormat f = new SimpleDateFormat("yyyyMMddHHmmss");

Random r = new Random();

int x = r.nextInt(9999);

String str = String.format("%04d", x);

orderid = Long.parseLong(f.format(time) + str);

}

public Order(int id) {

orderid = id;

}

public void setAddress(String a) {

addr = a;

}

public String getAddress() {

return addr;

}

public void setAmount(double a) {

amount = a;

}

public double getAmount() {

return amount;

}

public void setBussinessId(String a) {

bussinessid = a;

}

public String getBussinessId() {

return bussinessid;

}

public void setCommodityId(String[] a) {

StringBuffer sb = new StringBuffer();

for (int i = 0; i < a.length; i++) {

// System.out.println(a[i]);

if (i == a.length - 1) {

sb.append(a[i]);

} else {

sb.append(a[i] + ",");

}

}

commodityid = new String(sb.toString());

// System.out.println(commodityid);

}

public void setCommodityId(String a) {

commodityid = new String(a);

}

public String getCommodityId() {

return commodityid;

}

public String[] getCommodityIds() {

String[] a = commodityid.split(",");

return a;

}

public void setCustomId(String a) {

customerid = a;

}

public String getCustomId() {

return customerid;

}

public void setPhone(String a) {

phone = a;

}

public String getPhone() {

return phone;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public void setTime(Timestamp t) {

this.time = t;

}

public void setTime() {

date = new java.util.Date();

time = new Timestamp(date.getTime());

}

public Timestamp getTime() {

return time;

}

private boolean check(Connection con) {

boolean f = false;

PreparedStatement stm = null;

ResultSet rs = null;

try {

String sql;

f = false;

sql = "select \* from OrderTable where orderid=?";

stm = con.prepareStatement(sql);

stm.setLong(1, orderid);

rs = stm.executeQuery();

if (rs.next()) {

f = false;

} else {

f = true;

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

return f;

}

public int addToDB() {

Connection con = null;

PreparedStatement stm = null;

try {

con = DBConnect.connect();

while (!check(con)) {

setOrderid();

}

String sql;

sql = "insert into OrderTable values(?, ?, ?, ?, ?, ?, ?, ?, ?)";

stm = con.prepareStatement(sql);

stm.setString(1, addr);

stm.setDouble(2, amount);

stm.setString(3, bussinessid);

stm.setString(4, customerid);

stm.setString(5, commodityid);

stm.setTimestamp(6, time);

stm.setString(7, status);

stm.setString(8, phone);

stm.setLong(9, orderid);

stm.executeUpdate();

DBConnect.disconnect(con, stm);

return 1;

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(Order.class.getName()).log(Level.SEVERE, null, ex);

}

}

return 1;

}

public void deleteFromDB() {

Connection con = DBConnect.connect();

PreparedStatement stm = null;

try {

String sql;

sql = "delete from OrderTable where orderid = ?";

stm = con.prepareStatement(sql);

System.out.println(getOrderid());

stm.setString(1, String.valueOf(this.getOrderid()));

stm.executeUpdate();

} catch (SQLException ex) {

Logger.getLogger(Order.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(Order.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public void updateToDB() {

Connection con = DBConnect.connect();

PreparedStatement stm = null;

try {

String sql;

sql = "update OrderTable set addr=?, amount=?, bussinessid=?, customerid=?, commodityid=?, datetime=?,"

+ " status=?, phone=? where orderid=?";

stm = con.prepareStatement(sql);

stm.setString(1, addr);

stm.setDouble(2, amount);

stm.setString(3, bussinessid);

stm.setString(4, customerid);

stm.setString(5, commodityid);

stm.setTimestamp(6, time);

stm.setString(7, status);

stm.setString(8, phone);

stm.setLong(9, orderid);

stm.executeUpdate();

} catch (SQLException ex) {

Logger.getLogger(Order.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(Order.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

**PJFont.java**

package fattydelivery.lib;

import java.awt.Font;

/\*\*

\*

\* @author Administrator

\*/

public class PJFont {

public final static Font ft = new Font("TrueType", Font.PLAIN, 20);

}

**User.java**

package fattydelivery.lib;

import java.sql.\*;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author Administrator

\*/

public class User {

@Override

public String toString() {

return "User{" + "username=" + username + ", password=" + password + ", gender=" + gender + ", phone=" + phone + ", email=" + email + ", dob=" + dob + ", addr=" + addr + ", regdate=" + regdate + ", regtime=" + regtime + ", usertype=" + usertype + '}';

}

private String username;

private String password;

private String gender;

private String phone;

private String email;

private java.util.Date dob;

private String addr;

private java.util.Date regdate;

private Timestamp regtime;

private int usertype;

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender.toLowerCase();

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Date getDob() {

return dob;

}

public void setDob(String a) {

Date s = null;

try {

s = new SimpleDateFormat("yyyy-MM-dd").parse(a);

} catch (ParseException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

this.dob = s;

}

public void setDob(Date dob) {

this.dob = dob;

}

public String getAddr() {

return addr;

}

public void setAddr(String addr) {

this.addr = addr;

}

public int getUsertype() {

return usertype;

}

public void setUsertype(int usertype) {

this.usertype = usertype;

}

public void setRegTime() {

regdate = new java.util.Date();

regtime = new Timestamp(regdate.getTime());

}

public Timestamp getRegTime() {

return regtime;

}

public void setUsername(String a) {

username = a;

}

public String getUsername() {

return username;

}

public void setPassword(String a) {

password = a;

}

public String getPassword() {

return password;

}

private boolean check(Connection con) {

boolean f = false;

PreparedStatement stm = null;

ResultSet rs = null;

try {

String sql;

f = false;

if (usertype == 1) {

sql = "select \* from CustomerTable where username=?";

} else {

sql = "select \* from BussinessTable where username=?";

}

stm = con.prepareStatement(sql);

stm.setString(1, username);

rs = stm.executeQuery();

if (rs.next()) {

f = false;

} else {

f = true;

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

return f;

}

public boolean checkPassword() {

boolean f = false;

Connection con = DBConnect.connect();

PreparedStatement stm = null;

ResultSet rs = null;

try {

String sql;

f = false;

if (usertype == 1) {

sql = "select \* from CustomerTable where username=? and password=?";

} else {

sql = "select \* from BussinessTable where username=? and password=?";

}

stm = con.prepareStatement(sql);

stm.setString(1, username);

stm.setString(2, password);

rs = stm.executeQuery();

if (rs.next()) {

f = true;

} else {

f = false;

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

return f;

}

public boolean checkUesrname() {

Connection con = DBConnect.connect();

boolean f = !check(con);

DBConnect.disconnect(con);

return f;

}

public int addToDB() {

try {

Connection con;

con = DBConnect.connect();

if (!check(con)) {

return 0;

}

String sql;

if (usertype == 1) {

sql = "insert into CustomerTable values(?, ?, ?, ?, ?, ?, ?, ?);";

} else {

sql = "insert into BussinessTable values(?, ?, ?, ?, ?, ?, ?, ?);";

}

PreparedStatement stm = con.prepareStatement(sql);

stm.setString(1, username);

stm.setString(2, password);

stm.setString(3, gender);

stm.setString(4, phone);

stm.setString(5, email);

if (dob != null) {

stm.setDate(6, new java.sql.Date(dob.getTime()));

} else {

stm.setDate(6, null);

}

stm.setString(7, addr);

stm.setTimestamp(8, regtime);

stm.executeUpdate();

DBConnect.disconnect(con, stm);

return 1;

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

return 1;

}

public void updateToDB() {

Connection con = null;

con = DBConnect.connect();

String sql = null;

if (usertype == 1) {

sql = "update CustomerTable set password=?, gender=?, phone=?, email=?, dob=?, addr=? where username=?";

} else {

sql = "update BussinessTable set password=?, gender=?, phone=?, email=?, dob=?, addr=? where username=?";

}

try {

PreparedStatement stm = con.prepareStatement(sql);

stm.setString(1, password);

stm.setString(2, gender);

stm.setString(3, phone);

stm.setString(4, email);

if (dob != null) {

stm.setDate(5, new java.sql.Date(dob.getTime()));

} else {

stm.setDate(5, null);

}

stm.setString(6, addr);

stm.setString(7, username);

stm.executeUpdate();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

public void getFromDB() {

Connection con = DBConnect.connect();

String sql = null;

if (usertype == 1) {

sql = "select \* from CustomerTable where username=?";

} else {

sql = "select \* from BussinessTable where username=?";

}

PreparedStatement stm = null;

try {

stm = con.prepareStatement(sql);

stm.setString(1, username);

// System.out.println(stm);

ResultSet rs = stm.executeQuery();

if (rs.next()) {

password = rs.getString(2);

gender = rs.getString(3);

phone = rs.getString(4);

email = rs.getString(5);

dob = new java.util.Date(rs.getDate(6).getTime());

addr = rs.getString(7);

regtime = rs.getTimestamp(8);

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

public Order[] getAllOrder() {

Connection con = DBConnect.connect();

PreparedStatement stm = null;

ResultSet rs = null;

String sql = null;

Vector vec = new Vector<Order>();

Order[] o = null;

try {

if (usertype == 1) {

sql = "select \* from OrderTable where customerid=?";

} else {

sql = "select \* from OrderTable where bussinessid=?";

}

stm = con.prepareStatement(sql);

stm.setString(1, username);

rs = stm.executeQuery();

for (; rs.next();) {

Order tmp = new Order();

tmp.setAddress(rs.getString(1));

tmp.setAmount(rs.getDouble(2));

tmp.setBussinessId(rs.getString(3));

tmp.setCustomId(rs.getString(4));

tmp.setCommodityId(rs.getString(5));

tmp.setTime(new Timestamp(rs.getTimestamp(6).getTime()));

tmp.setStatus(rs.getString(7));

tmp.setPhone(rs.getString(8));

tmp.setOrderid(rs.getLong(9));

vec.add(tmp);

}

o = new Order[vec.size()];

for (int i = 0; i < vec.size(); i++) {

o[i] = (Order) vec.get(i);

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

con.close();

} catch (SQLException ex1) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex1);

}

}

return o;

}

public Commodity[] getAllCommodity() {

PreparedStatement stm = null;

String sql = null;

Connection con = null;

ResultSet rs = null;

Commodity[] com = null;

if (this.usertype == 2) {

try {

con = DBConnect.connect();

sql = "select \* from CommodityTable where bussinessid = ?";

stm = con.prepareStatement(sql);

stm.setString(1, username);

rs = stm.executeQuery();

Vector vec = new Vector();

for (; rs.next();) {

Commodity c = new Commodity();

c.setId(rs.getString(1));

c.setCost(rs.getDouble(2));

c.setBussinessId(rs.getString(3));

c.setImg(rs.getBinaryStream(4));

c.setImgext(rs.getString(5));

vec.add(c);

}

com = new Commodity[vec.size()];

for (int i = 0; i < vec.size(); i++) {

com[i] = (Commodity) vec.get(i);

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

} else {

try {

con = DBConnect.connect();

sql = "select \* from CommodityTable";

stm = con.prepareStatement(sql);

rs = stm.executeQuery();

Vector vec = new Vector();

for (; rs.next();) {

Commodity c = new Commodity();

c.setId(rs.getString(1));

c.setCost(rs.getDouble(2));

c.setBussinessId(rs.getString(3));

c.setImg(rs.getBinaryStream(4));

c.setImgext(rs.getString(5));

vec.add(c);

}

com = new Commodity[vec.size()];

for (int i = 0; i < vec.size(); i++) {

com[i] = (Commodity) vec.get(i);

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return com;

}

public static Commodity[] getAllCommodities() {

PreparedStatement stm = null;

String sql = null;

Connection con = null;

ResultSet rs = null;

Commodity[] com = null;

try {

con = DBConnect.connect();

sql = "select \* from CommodityTable";

stm = con.prepareStatement(sql);

rs = stm.executeQuery();

Vector vec = new Vector();

for (; rs.next();) {

Commodity c = new Commodity();

c.setId(rs.getString(1));

c.setCost(rs.getDouble(2));

c.setBussinessId(rs.getString(3));

c.setImg(rs.getBinaryStream(4));

c.setImgext(rs.getString(5));

vec.add(c);

}

com = new Commodity[vec.size()];

for (int i = 0; i < vec.size(); i++) {

com[i] = (Commodity) vec.get(i);

}

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

} finally {

try {

rs.close();

stm.close();

con.close();

} catch (SQLException ex) {

Logger.getLogger(User.class.getName()).log(Level.SEVERE, null, ex);

}

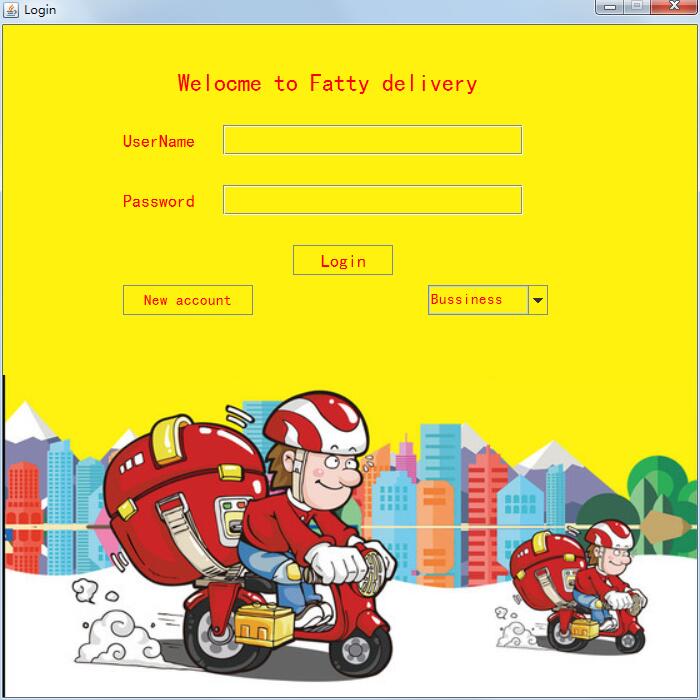
}

return com;

}

}

**fattydelivery.log**

**LoginFrame.java**

package fattydelivery.log;

import fattydelivery.bussiness.\*;

import fattydelivery.customer.\*;

import fattydelivery.lib.\*;

import com.mysql.jdbc.PreparedStatement;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.SQLException;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

public class LoginFrame extends JFrame implements ActionListener {

JPanel jp;

JLabel UserName, Password, jl, j4;

JTextField jta1;

Icon i;

JButton jb, jb1;

JComboBox jcb;

JPasswordField jpf;

User u;

String str, regex;

Pattern pattern;

Matcher matcher;

public LoginFrame() {

UIManager.put("Button.font", PJFont.ft);

UIManager.put("Label.font", PJFont.ft);

UIManager.put("TextField.font", PJFont.ft);

UIManager.put("TextArea.font", PJFont.ft);

UIManager.put("ComboBox.font", PJFont.ft);

UIManager.put("PasswordField.font", PJFont.ft);

UIManager.put("RadioButton.font", PJFont.ft);

UIManager.put("TabbedPane.font", PJFont.ft);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

u = new User();

jp = new JPanel();

jp.setLayout(null);

jp.setBackground(new Color(255, 242, 016));

jta1 = new JTextField(10);

jta1.setBounds(220, 100, 300, 30);

jta1.setBackground(new Color(255, 242, 016));

jp.add(jta1);

jpf = new JPasswordField();

jpf.setBounds(220, 160, 300, 30);

jpf.setBackground(new Color(255, 242, 016));

jp.add(jpf);

jl = new JLabel();

i = new ImageIcon("img\\picture1.png");

jl.setIcon(i);

jl.setBounds(0, 350, i.getIconWidth(), i.getIconHeight());

jp.add(jl, new Integer(Integer.MIN\_VALUE));

UserName = new JLabel("UserName");

Font f1 = new Font("黑体", Font.PLAIN, 18);

UserName.setFont(f1);

UserName.setOpaque(true);

UserName.setBackground(new Color(255, 242, 016));

UserName.setForeground(Color.red);

UserName.setBounds(120, 100, 80, 30);

jp.add(UserName);

Password = new JLabel("Password");

Password.setFont(f1);

Password.setOpaque(true);

Password.setBackground(new Color(255, 242, 016));

Password.setForeground(Color.red);

Password.setBounds(120, 160, 80, 30);

jp.add(Password);

Font f2 = new Font("黑体", Font.PLAIN, 24);

j4 = new JLabel("Welocme to Fatty delivery");

j4.setFont(f2);

j4.setOpaque(true);

j4.setBackground(new Color(255, 242, 016));

j4.setForeground(Color.red);

j4.setBounds(175, 40, 350, 35);

jp.add(j4);

jb = new JButton("Login");

Font f3 = new Font("黑体", Font.PLAIN, 15);

jb.setFont(f1);

jb.setOpaque(true);

jb.setBackground(new Color(255, 242, 016));

jb.setForeground(Color.red);

jb.setBounds(290, 220, 100, 30);

jb.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

u.setUsername(jta1.getText());

u.setPassword(new String(jpf.getPassword()));

str = jta1.getText();

regex = "^[a-zA-Z0-9\_]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(jp, "Username must not be special character.");

return;

}

str = String.valueOf(jpf.getPassword());

regex = "^[a-zA-Z0-9]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(jp, "Password must not be special character.");

return;

}

if (jcb.getSelectedIndex() == 0) {

u.setUsertype(2);

} else {

u.setUsertype(1);

}

// System.out.println(u.getUsertype());

if (jta1.getText().length() == 0) {

JOptionPane.showMessageDialog(jp, "Empty UserName", "Warning", JOptionPane.WARNING\_MESSAGE);

} else if (jpf.getPassword().length == 0) {

JOptionPane.showMessageDialog(jp, "Empty Password", "Warning", JOptionPane.WARNING\_MESSAGE);

} else if (u.checkPassword()) {

System.out.println("Success");

u.getFromDB();

if (u.getUsertype() == 1) {

new CustomerFrame(u);

} else {

new BussinessFrame(u);

}

dispose();

} else {

JOptionPane.showMessageDialog(jp, "Incorrect User Name or Password", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

});

jp.add(jb);

jb1 = new JButton("New account");

jb1.setFont(f3);

jb1.setOpaque(true);

jb1.setBackground(new Color(255, 242, 016));

jb1.setForeground(Color.red);

jb1.setBounds(120, 260, 130, 30);

jb1.addActionListener(this);

jp.add(jb1);

jcb = new JComboBox();

jcb.addItem("Bussiness");

jcb.addItem("Customer");

jcb.setFont(f3);

jcb.setOpaque(true);

jcb.setBackground(new Color(255, 242, 016));

jcb.setForeground(Color.red);

jcb.setBounds(425, 260, 120, 30);

jp.add(jcb);

add(jp);

setTitle("Login");

setVisible(true);

setSize(700, 700);

setResizable(false);

this.setLocationRelativeTo(null);

}

/\*

\* public static void main(String args[]) { LoginFrame FD=new LoginFrame();

}

\*/

// Variables declaration - do not modify

// End of variables declaration

@Override

public void actionPerformed(ActionEvent e) {

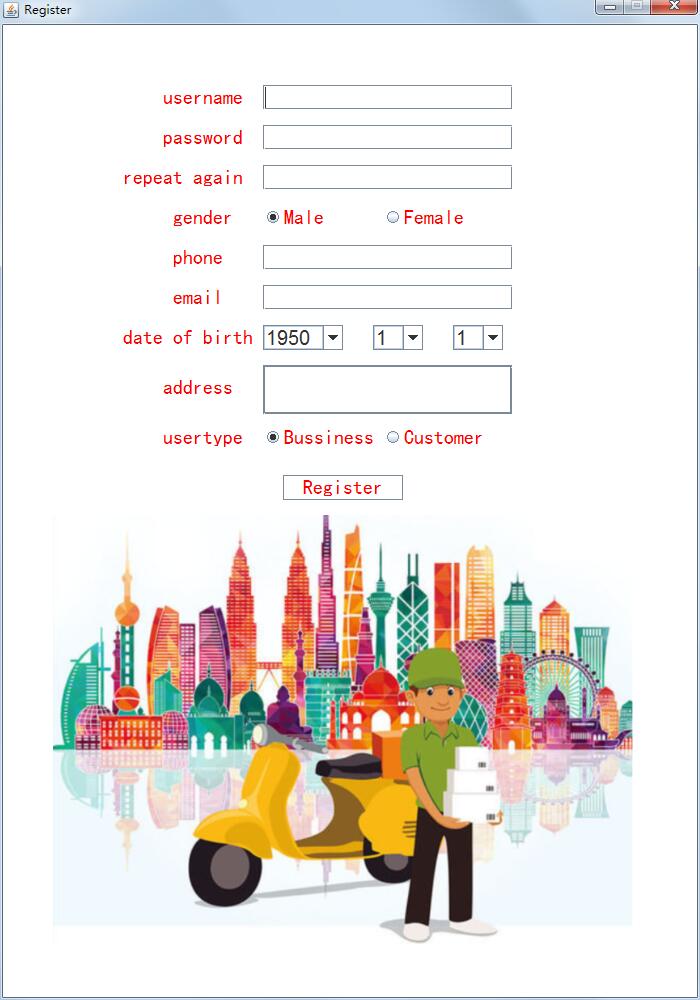
Register r = new Register();

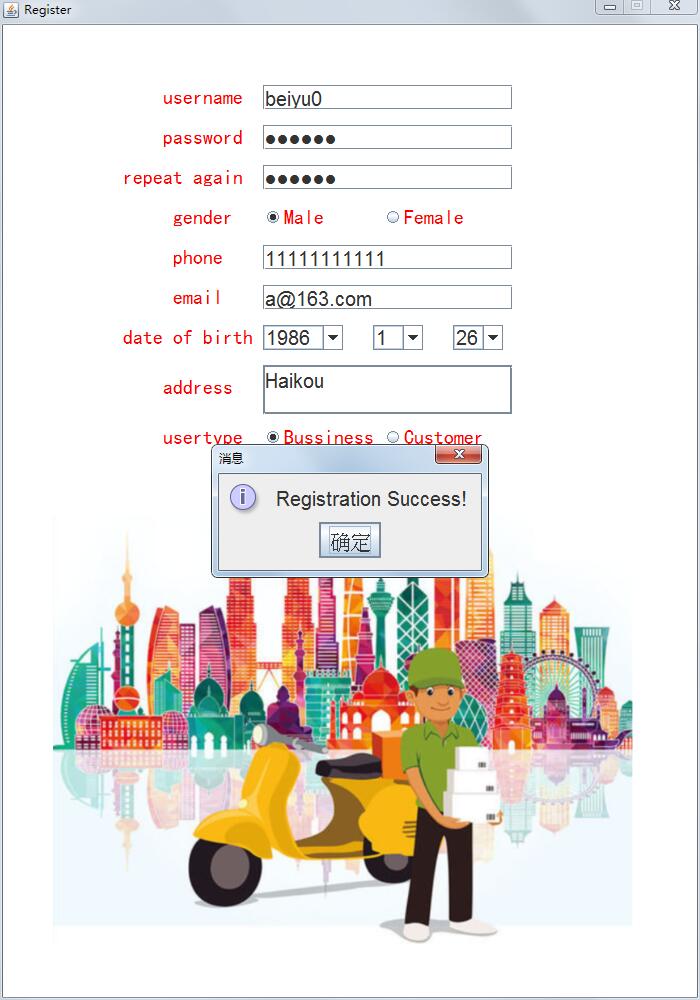
this.dispose();

}

}

**Register.java**

****

****

package fattydelivery.log;

import fattydelivery.lib.\*;

import com.mysql.jdbc.PreparedStatement;

import java.sql.SQLException;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.\*;

public class Register extends javax.swing.JFrame implements ActionListener {

JPanel jp;

JLabel jl1, jl2, jl3, jl4, jl5, jl6, jl7, jl9;

JTextField jtf1, jtf3, jtf4, jtf5, jtf7;

JPasswordField jpf;

JTextArea jtr;

Icon i;

JRadioButton jrb1, jrb2;

ButtonGroup bg;

JButton jb;

JComboBox JCBdoby;

JComboBox JCBdobm;

JComboBox JCBdobd;

JLabel JLusertype, JLrpassword;

JPasswordField JPFrpassword;

JRadioButton JRBcustomer, JRBbussiness;

ButtonGroup BG;

User u;

String str, regex;

Pattern pattern;

Matcher matcher;

public Register() {

u = new User();

jp = new JPanel();

jp.setLayout(null);

jp.setBackground(Color.WHITE);

jl1 = new JLabel("username");

Font f1 = new Font("黑体", Font.PLAIN, 20);

jl1.setFont(f1);

jl1.setOpaque(true);

jl1.setBackground(Color.WHITE);

jl1.setForeground(Color.red);

jl1.setBounds(160, 60, 80, 25);

jp.add(jl1);

jl2 = new JLabel("password");

jl2.setFont(f1);

jl2.setOpaque(true);

jl2.setBackground(Color.WHITE);

jl2.setForeground(Color.red);

jl2.setBounds(160, 40 + 60, 80, 25);

jp.add(jl2);

JLrpassword = new JLabel("repeat again");

JLrpassword.setFont(f1);

JLrpassword.setOpaque(true);

JLrpassword.setBackground(Color.WHITE);

JLrpassword.setForeground(Color.red);

JLrpassword.setBounds(120, 80 + 60, 120, 25);

jp.add(JLrpassword);

jl3 = new JLabel("gender");

jl3.setFont(f1);

jl3.setOpaque(true);

jl3.setBackground(Color.WHITE);

jl3.setForeground(Color.red);

jl3.setBounds(170, 80 + 100, 60, 25);

jp.add(jl3);

jl4 = new JLabel("phone");

jl4.setFont(f1);

jl4.setOpaque(true);

jl4.setBackground(Color.WHITE);

jl4.setForeground(Color.red);

jl4.setBounds(170, 120 + 100, 60, 25);

jp.add(jl4);

jl5 = new JLabel("email");

jl5.setFont(f1);

jl5.setOpaque(true);

jl5.setBackground(Color.WHITE);

jl5.setForeground(Color.red);

jl5.setBounds(170, 160 + 100, 60, 25);

jp.add(jl5);

jl6 = new JLabel("date of birth");

jl6.setFont(f1);

jl6.setOpaque(true);

jl6.setBackground(Color.WHITE);

jl6.setForeground(Color.red);

jl6.setBounds(120, 200 + 100, 130, 25);

jp.add(jl6);

jl7 = new JLabel("address");

jl7.setFont(f1);

jl7.setOpaque(true);

jl7.setBackground(Color.WHITE);

jl7.setForeground(Color.red);

jl7.setBounds(160, 250 + 100, 80, 25);

jp.add(jl7);

JLusertype = new JLabel("usertype");

JLusertype.setFont(f1);

JLusertype.setOpaque(true);

JLusertype.setBackground(Color.WHITE);

JLusertype.setForeground(Color.red);

JLusertype.setBounds(160, 300 + 100, 80, 25);

jp.add(JLusertype);

jtf1 = new JTextField();

jtf1.setBounds(260, 0 + 60, 250, 25);

jtf1.setBackground(Color.WHITE);

jp.add(jtf1);

jpf = new JPasswordField();

jpf.setBounds(260, 40 + 60, 250, 25);

jpf.setBackground(Color.WHITE);

jp.add(jpf);

JPFrpassword = new JPasswordField();

JPFrpassword.setBounds(260, 80 + 60, 250, 25);

JPFrpassword.setBackground(Color.WHITE);

jp.add(JPFrpassword);

jrb1 = new JRadioButton("Male");

jrb1.setFont(f1);

jrb1.setOpaque(true);

jrb1.setBackground(Color.WHITE);

jrb1.setForeground(Color.red);

jrb1.setBounds(260, 80 + 100, 80, 25);

jrb1.setSelected(true);

jp.add(jrb1);

jrb2 = new JRadioButton("Female");

jrb2.setFont(f1);

jrb2.setOpaque(true);

jrb2.setBackground(Color.WHITE);

jrb2.setForeground(Color.red);

jrb2.setBounds(380, 80 + 100, 100, 25);

jp.add(jrb2);

jtf3 = new JTextField();

jtf3.setBounds(260, 120 + 100, 250, 25);

jtf3.setBackground(Color.WHITE);

jp.add(jtf3);

jtf4 = new JTextField();

jtf4.setBounds(260, 160 + 100, 250, 25);

jtf4.setBackground(Color.WHITE);

jp.add(jtf4);

jtr = new JTextArea();

// jtr.setBounds(260, 240+100, 250, 50);

jtr.setBackground(Color.WHITE);

jtr.setBorder(BorderFactory.createLineBorder(new Color(122, 138, 153)));

jtr.setLineWrap(true);

JScrollPane JSP = new JScrollPane(jtr);

JSP.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

JSP.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

// JTAaddr.setBounds(150, 10+(MHEIGHT+DELTAY)\*7, MWIDTH2+20, MHEIGHT\*5);

JSP.setBounds(260, 240 + 100, 250, 50);

jp.add(JSP);

JCBdoby = new JComboBox();

for (int i = 1950; i <= 2019; i++) {

JCBdoby.addItem(Integer.toString(i));

}

JCBdoby.setBounds(260, 200 + 100, 80, 25);

JCBdoby.setBackground(Color.WHITE);

jp.add(JCBdoby);

JCBdobm = new JComboBox();

for (int i = 1; i <= 12; i++) {

JCBdobm.addItem(Integer.toString(i));

}

JCBdobm.setBounds(370, 200 + 100, 50, 25);

JCBdobm.setBackground(Color.WHITE);

jp.add(JCBdobm);

JCBdobd = new JComboBox();

for (int i = 1; i <= 31; i++) {

JCBdobd.addItem(Integer.toString(i));

}

JCBdobd.setBounds(450, 200 + 100, 50, 25);

JCBdobd.setBackground(Color.WHITE);

jp.add(JCBdobd);

bg = new ButtonGroup();

bg.add(jrb1);

bg.add(jrb2);

JRBbussiness = new JRadioButton("Bussiness");

JRBbussiness.setFont(f1);

JRBbussiness.setOpaque(true);

JRBbussiness.setBackground(Color.WHITE);

JRBbussiness.setForeground(Color.red);

JRBbussiness.setBounds(260, 300 + 100, 120, 25);

JRBbussiness.setSelected(true);

jp.add(JRBbussiness);

JRBcustomer = new JRadioButton("Customer");

JRBcustomer.setFont(f1);

JRBcustomer.setOpaque(true);

JRBcustomer.setBackground(Color.WHITE);

JRBcustomer.setForeground(Color.red);

JRBcustomer.setBounds(380, 300 + 100, 120, 25);

jp.add(JRBcustomer);

BG = new ButtonGroup();

BG.add(JRBbussiness);

BG.add(JRBcustomer);

jl9 = new JLabel();

jl9 = new JLabel();

i = new ImageIcon("img\\picture2.png");

jl9.setIcon(i);

jl9.setBounds(50, 350 + 140, i.getIconWidth(), i.getIconHeight());

jp.add(jl9, new Integer(Integer.MIN\_VALUE));

jb = new JButton("Register");

jb.setFont(f1);

jb.setOpaque(true);

jb.setBackground(Color.WHITE);

jb.setForeground(Color.red);

jb.setBounds(280, 310 + 140, 120, 25);

jp.add(jb);

jb.addActionListener(this);

User u = new User();

add(jp);

setTitle("Register");

setVisible(true);

setSize(700, 1000);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

this.setResizable(false);

this.setLocationRelativeTo(null);

this.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

super.windowClosing(e);

new LoginFrame();

}

});

}

@Override

public void actionPerformed(ActionEvent e) {

u.setUsername(jtf1.getText());

u.setPassword(new String(jpf.getPassword()));

if (jrb1.isSelected()) {

u.setGender("male");

} else {

u.setGender("female");

}

u.setPhone(jtf3.getText());

u.setEmail(jtf4.getText());

u.setDob(JCBdoby.getSelectedItem().toString() + "-" + JCBdobm.getSelectedItem().toString() + "-" + JCBdobd.getSelectedItem().toString());

u.setAddr(jtr.getText());

u.setRegTime();

if (JRBcustomer.isSelected()) {

u.setUsertype(1);

} else {

u.setUsertype(2);

}

// System.out.println("???");

if (jtf1.getText().length() == 0) {

JOptionPane.showMessageDialog(jp, "Username must not be empty.");

return;

}

// System.out.println("username ok");

String s1 = new String(jpf.getPassword()), s2 = new String(JPFrpassword.getPassword());

if (!s1.equals(s2)) {

JOptionPane.showMessageDialog(jp, "Inconsistent password input twice.");

return;

}

// System.out.println("password ok");

if (jtf3.getText().length() == 0) {

JOptionPane.showMessageDialog(jp, "Phone number must not be empty.");

return;

}

// System.out.println("phone ok");

if (jtr.getText().length() == 0) {

JOptionPane.showMessageDialog(jp, "Address must not be empty.");

return;

}

// System.out.println("addr ok");

if (u.checkUesrname()) {

JOptionPane.showMessageDialog(jp, "Username already exists.");

return;

}

// System.out.println("user ok");

str = jtf1.getText();

regex = "^[a-zA-Z0-9\_]+$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(jp, "Username must not be special character or Hanzi");

return;

}

str = String.valueOf(jpf.getPassword());

// regex = "^[a-zA-Z0-9]+$";

// pattern = Pattern.compile(regex);

// matcher = pattern.matcher(str);

// if (!matcher.matches()) {

// JOptionPane.showMessageDialog(jp, "Password must not be special character.");

// return;

// }

str = jtf3.getText();

regex = "^[0-9]{11}$";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(jp, "Telephone number must be 11 digits.");

return;

}

str = jtf4.getText();

regex = "[a-zA-Z-]{0,}[0-9]{0,}@(([a-zA-Z0-9]-\*){1,}\\.){1,3}[a-zA-Z\\-]{1,}";

pattern = Pattern.compile(regex);

matcher = pattern.matcher(str);

if (!matcher.matches()) {

JOptionPane.showMessageDialog(jp, "The format of email is incorrect.");

return;

}

u.addToDB();

JOptionPane.showMessageDialog(jp, "Registration Success!");

this.dispose();

new LoginFrame();

}

}