|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»**  **Тема: «Графический пользовательский интерфейс с использованием библиотеки swing»** | |
|  | |
| Выполнил студент группы ИКБО-07-18 | Мишанов А.Б. |
| Принял преподаватель | Колесников С.Н. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись руководителя)* |

**Задание 1**

Разработать графический интерфейс приложения с использованием библиотеки swing

**Требования к заданию:**

1. Классы форм и классы, реализующие бизнес логику, должны быть в отдельных классах.
2. Использовать файлы для хранения сериализованных объектов.
3. Справочные данные в формах ввода должны выбираться средствами элемента JComboBox источником элементов которого может выть список или массив, заливаемый из соответствующего файла. Для справочников, в которых количество значений небольшое можно использовать внутренний список, который заполняется в коде.
4. Должна быть предусмотрена форма авторизации, позволяющая выполнять регистрацию (при условии, что это допускает логика приложения) и авторизацию.

Задачи

* разработка проекта пользовательского интерфейса и его представление в форме дерева связей между элементами (окнами);
* реализация интерфейса средствами swing, используя оба подхода: программирование интерфейса путем создания кода классов форм вручную и средствами визуального GUI редактора.

Задание к лабораторной работе

1. Для каждого класса лабораторной 3 разработать проект формы заполнения, модификации, отображения свойств экземпляра класса и удаления экземпляра.
2. Разработать проекты форм для демонстрации выполнения функционала программы, лабораторной работы 3.
3. Разработать проект дерева интерфейса приложения.
4. Реализовать формы для ввода и вывода свойств одного объекта, предусмотрев:

* использование списковых элементов управления для заполнения отдельных свойств объекта класса;
* контроль достоверности вводимых данных.

1. Разработать форму для заполнения экземпляров класса из файла. Используйте класс, который был выбран в лабораторной работе 4. Предусмотрите сохранение данных объекта в файл и отображение объектов из файла в элементе JTable.

При выполнении открытия и сохранения файлов используйте компонент JFileChooser.

1. Реализовать форму управления приложением - тестирование функционала задания лабораторной работы 3.

Требования выполнению

1. Две формы должны быть разработаны без использования редактора GUI, остальные либо JFrame.
2. Основной фон форм – светло серый.
3. Текст элементов управления должен быть оформлен: шрифт – Times New Rоman, размер14, обычный, синий, выравнивание меток - по левому краю, в текстовом поле: числовые значения по правому краю, строковые полевому. Надписи на кнопках по центру.
4. Способ размещения элементов на форме выбрать самостоятельно.
5. Приложение должно иметь главную форму – первый уровень дерева интерфейса.
6. Вход пользователей в приложение по авторизации. Данные пользователей должны храниться в файле, лучше в форме объектов.
7. Предусмотреть обработку исключительных ситуаций через аппарат исключений.
8. Для представления сообщений использовать диалоговые окна сообщений.

**Тестирование:**

В качестве тестирования приведены скриншоты всех окон:

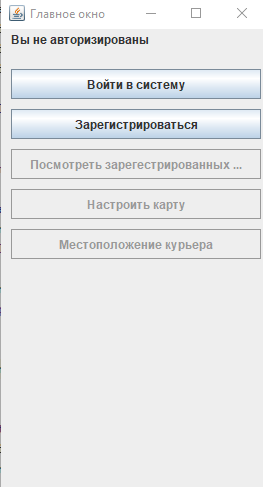


Рисунок 1 – Главное окно

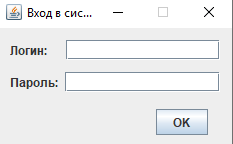


Рисунок 2 – Окно входа в систему

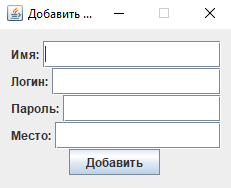


Рисунок 3 – Окно регистрации/добавления пользователя

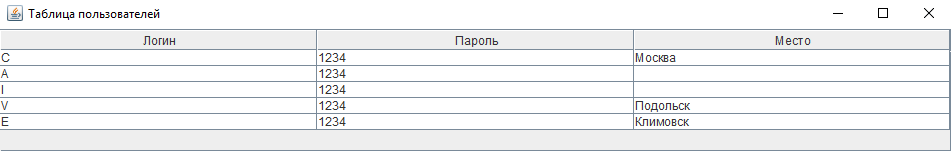


Рисунок 4 – Окно таблицы зарегестрированных пользователей

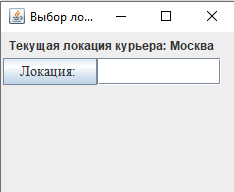


Рисунок 5 – Окно выбора локации курьера

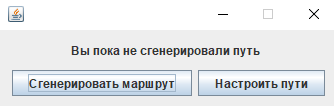


Рисунок 6 – окно настройки карты

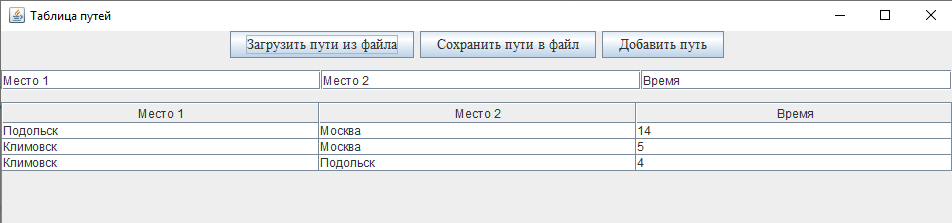
****

Рисунок 7 – Окно таблицы путей

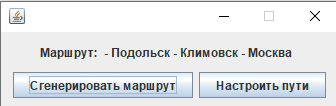
****

Рисунок 8 – окно настройки карты с учетом того, что клиент расположен в Подольск, а Клиент - в Москве

**Исходный код**

Класс User

**package** com.company;  
  
**import** java.io.Serializable;  
  
**public class** User **implements** Serializable {  
 **private** String **name**;  
 **private** String **login**;  
 **private** String **password**;  
 User(){  
 **name**=**null**;**login**=**null**;**password**=**null**;  
 }  
 User(String name, String login, String password){  
 **this**.**name**=name; **this**.**login**=login; **this**.**password**=password;  
 }  
 **public boolean** enter(String login, String password){  
 **return** (**this**.**login**==login & **this**.**password**==password);  
 }  
 **public** String getLogin() {  
 **return login**;  
 }  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public** String getPlace(){ **return null**; }  
}  
  
**class** Client **extends** User{  
 **private** String **place**;  
 Client(String n, String l, String p, String place){  
 **super**(n,l,p);  
 **this**.**place**=place;  
 }  
 @Override  
 **public** String getPlace(){  
 **return place**;  
 }  
 **public void** setPlace(String s){  
 **place**=s;  
 }  
}

*//this.Out();* }  
}

Класс Map

**package** com.company;  
  
**import** java.awt.\*;  
  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.io.\*;  
**import** java.util.HashMap;  
  
**public class** Map **implements** Serializable{  
 *//Вложенный класс путь* **static public class** Path **implements** Serializable{  
 **private** String [] **points**;  
 **private int time**;  
 Path(String points[],**int** time){  
 **this**.**points**=points;  
 **this**.**time**=time;  
 }  
 **public** String showPath(){  
 **return** (**points**[0]+**" "**+**points**[1]+**" "**+**time**);  
 }  
 **public** String[] getPoints(){  
 **return points**;  
 }  
 **public** String getPlace1(){  
 **return points**[0];  
 }  
 **public** String getPlace2(){  
 **return points**[1];  
 }  
 **public int** getTime() {  
 **return time**;  
 }  
 }  
  
 **private** ArrayList<Path> **paths**;  
 **private** ArrayList **points**;  
 **private int ways**[][];  
 Map(){  
 **paths**=**new** ArrayList<>(); *//массив путей* **points**=**new** ArrayList(); *//массив мест* **ways**=**new int**[100][100]; *//матрица смежности путей. Отражает, есть ли путь(ребро) между двумя точками. Если путя нет, то равняется бесконечности (в рамках программы - Max\_int)* **for** (**int** i=0;i<100;i++)  
 **for**(**int** j=0;j<100;j++)  
 **ways**[i][j]=Integer.***MAX\_VALUE***;  
 }  
 **public void** addPath(String place1,String place2,**int** time){  
 String[] arr=**new** String[2];  
 arr[0]=place1; arr[1]=place2;  
 Path path=**new** Path(arr,time);  
 **paths**.add(path);  
  
 }  
 **public void** setStuff(){  
 String[] tempStringArr = **new** String[2];  
 **for** (Path p : **paths** ) {  
 tempStringArr = p.getPoints();  
 *//Проверяем наличие точки в массиве мест, если этого места нет, то добавляем его* **if** (!**points**.contains(tempStringArr[0]))  
 **points**.add(tempStringArr[0]);  
 **if** (!**points**.contains(tempStringArr[1]))  
 **points**.add(tempStringArr[1]);  
 *//Добавляем путь в матрицу смежности* **ways**[**points**.indexOf(tempStringArr[0])][**points**.indexOf(tempStringArr[1])]=p.getTime();  
 **ways**[**points**.indexOf(tempStringArr[1])][**points**.indexOf(tempStringArr[0])]=p.getTime();  
  
 }  
 }  
  
 **public** ArrayList<Path> getPaths(){  
 **return paths**;  
 }  
  
 **public** ArrayList genShortWay(String place1, String place2){  
  
 **int** MAX=Integer.***MAX\_VALUE***; *//Бесконечность. Нужна для проверки наличия ребра* **int** pointCount=**points**.size(); *//Количетсво мест* **int** D[]=**new int**[pointCount]; *//массив меток мест* **int** P[]=**new int**[pointCount]; *//массив точек* **int** startPoint=**points**.indexOf(place1); *//стартовая точка* **boolean** []visited=**new boolean**[pointCount]; *//массив посещенных точек (индекс соответствует индексу точки в массиве points)  
 //Заполняем D, P и visited значениями по умолчанияю* Arrays.*fill*(D,MAX); Arrays.*fill*(P,-1);  
 Arrays.*fill*(visited,**false**);  
 D[startPoint]=0; *//устанавливаем метку стартовой точки за 0* **for** (**int** i=0;i<pointCount;i++)  
 D[i]=**ways**[startPoint][i]; *//устанавливаем метки на все остальные точки, на основе наличия расстояния до начальной метки (если прямого пути(ребра) нет, то бесконечность)* **int** index=0, u=0, min; *//index - индекс точки, метка которой имеет минимальное значение  
  
 //алгоритм Дейсктры* **for** (**int** i=0;i<pointCount;i++){  
 min=MAX;  
 **for** (**int** j=0;j<pointCount;j++){  
 **if** (!visited[j] && D[j]<min){  
 min=D[j]; index=j;  
 }  
 }  
 u=index;  
 visited[u]=**true**;  
 **for** (**int** j=0;j<pointCount;j++){  
 **if** (!visited[j] && **ways**[u][j]!=MAX && D[u]!=MAX && (D[u]+**ways**[u][j]<D[j])){  
 D[j]=D[u]+**ways**[u][j];  
 P[j]=u;  
 }  
 }  
 }  
 *//расложение маршрута* ArrayList stack=**new** ArrayList();  
 **int** target=**points**.indexOf(place2);  
 ArrayList res= **new** ArrayList();  
 **if** (D[**points**.indexOf(place2)] < Integer.***MAX\_VALUE***) {  
 **for** (**int** v = target; v != -1; v = P[v])  
 stack.add(0, v);  
 ArrayList sp = **new** ArrayList(stack.size());  
 **for** (**int** i = 0; i < stack.size(); i++)  
 sp.add(stack.get(i));  
  
 res.add(place1);  
 **for** (**int** i = 0; i < sp.size(); i++)  
 res.add(**points**.get((Integer) sp.get(i)));  
 }  
 **return** res;  
 }  
 **public** String show(){  
 String res=**""**;  
 **for** (**int** i=0; i<**paths**.size();i++){  
 res+=**paths**.get(i).showPath()+**"\n"**;  
 }  
 **return** res;  
 }  
 **public void** loadPaths(File f) {  
 **try**{  
 FileReader fr = **new** FileReader(f);  
 BufferedReader reader = **new** BufferedReader(fr);  
 String line;  
 String p1,p2;  
 **int** time;  
 **while** ((line = reader.readLine())!= **null**){  
 p1 = **new** String(line);  
 line = reader.readLine();  
 p2 = **new** String(line);  
 line = reader.readLine();  
 time = Integer.*parseInt*(line);  
 String p[] = {p1,p2};  
 **paths**.add(**new** Path(p, time));  
 }  
  
 }**catch** (Exception e){  
 System.***out***.println(e.toString());  
 }  
 }  
  
 **public void** setPaths(ArrayList<Path> paths) {  
 **this**.**paths** = paths;  
 }  
  
 **public** ArrayList getPoints() {  
 **return points**;  
 }  
  
 **public void** savePaths(File f){  
 **try**{  
 FileWriter fr = **new** FileWriter (f);  
 BufferedWriter writer = **new** BufferedWriter(fr);  
 String line;  
 String p1,p2;  
 **int** time;  
 **for** (Path p: **paths** ) {  
 writer.write(p.getPlace1()+**'\n'**);  
 writer.write(p.getPlace2()+**'\n'**);  
 Integer costil = p.getTime();  
 writer.write(costil.toString()+**'\n'**);  
 }  
 writer.close();  
 }**catch** (Exception e){  
 System.***out***.println(e.toString());  
 }  
 }  
}

Класс PizzaSytem

**package** com.company;  
  
**import** java.io.\*;  
**import** java.sql.SQLOutput;  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** PizzaSystem {  
 **private static** Map *map*; *//= new Map();* **private static** ArrayList<User> *users*; *//= new ArrayList<>();* **private static** User *currentUser*; *//= new User();* **private static** String *currentLocation*; *//= new String();* PizzaSystem(){  
 *// map = new Map();  
 //currentUser = new User();  
 //currentLocation = new String();  
 //users = new ArrayList<>();* }  
 **public static void** main(String[] args)**throws** IOException, ClassNotFoundException {  
 *users* = **new** ArrayList<>();  
 *map* = **new** Map();  
 *currentUser* = **new** User();  
 *currentLocation* = **""**;  
  
 }  
  
 **public void** setCurrentLocation(String currentLocation) {  
 PizzaSystem.*currentLocation* = currentLocation;  
 }  
  
 **public static** ArrayList<User> addUser(ArrayList<User> users, String name, String login, String password, String rep) {  
 users.add(**new** User(name, login, password));  
 **return** (users);  
 }  
  
 **public static** ArrayList<User> addClient(ArrayList<User> users, String name, String login, String password, String place) {  
 users.add(**new** Client(name, login, password, place));  
 **return** (users);  
 }  
 **public** String getCurrentLocation(){  
 **return** *currentLocation*;  
 }  
 **public static void** addPath(String p1, String p2, **int** time){  
 *map*.addPath(p1,p2,time);  
 }  
 **public void** addUser(User u){  
 **try** {  
 userLoad();  
 *users*.add(u);  
 userSave();  
 }**catch** (IOException | ClassNotFoundException e){  
 System.***out***.println(e.toString()+ **" Ой-ёй, что-то пошло не так"**);  
 }  
 }  
 **public** String getUser(){  
 **return** *currentUser*.getLogin()+**" Ваше местоположение: "**+*currentUser*.getPlace();  
 }  
 **public** String getLogin(){  
 **return** *currentUser*.getLogin();  
 }  
  
 **public boolean** findUser(String login, String password) {  
 **for** (User i : *users* ) {  
 String s1 = i.getLogin();  
 String s2 = i.getPassword();  
 **if** ((s1.equals(login)) && (s2.equals(password))) {  
 *currentUser* = i;  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
 **public** String generate(String yourPlace, String courierPlace){  
 *map*.setStuff();  
 ArrayList res = *map*.genShortWay(yourPlace,courierPlace);  
 String s = **"Маршрут: "**;  
 **for** (**int** i=0; i<res.size(); i++)  
 s += **" - "** + res.get(i);  
 **return** s;  
 }  
 **public static void** save()**throws** IOException {  
 File f = **new** File(**"map.txt"**);  
 f.createNewFile();  
 ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream(**"map.txt"**));  
 out.writeObject(*map*);  
  
 }  
 **public static void** load() **throws** IOException, ClassNotFoundException{  
 **try** {  
 File f = **new** File(**"map.txt"**);  
 ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream(**"map.txt"**));  
 *map* = (Map) in.readObject();  
 }**catch** (IOException | ClassNotFoundException e){  
 System.***out***.println(e.toString() + **" Что-то пошло не так"**);  
 }  
 }  
 **public void** userLoad() **throws** IOException, ClassNotFoundException{  
 **try**{  
 ObjectInputStream in = **new** ObjectInputStream(**new** FileInputStream(**"users.txt"**));  
 *users* = (ArrayList<User>) in.readObject();  
 in.close();;  
 }**catch** (IOException | ClassNotFoundException e){  
 System.***out***.println(e.toString() + **" Что-то пошло не так"**);  
 }  
 }  
 **public void** userSave() **throws** IOException{  
 ObjectOutputStream out = **new** ObjectOutputStream(**new** FileOutputStream(**"users.txt"**));  
 out.writeObject(*users*);  
 out.close();  
 }  
 **public** Map getMap() {  
 **return** *map*;  
 }  
 **public** ArrayList<Map.Path> getPaths(){  
 **return**(*map*.getPaths());  
 }  
 **public void** loadPaths(File f){  
 *map*.loadPaths(f);  
 }  
 **public void** setMap(Map m){  
 *map* = m;  
 }  
  
 **public** User getCurrentUser() {  
 **return** *currentUser*;  
 }  
  
 **public** ArrayList<User> getUsers() {  
 **return** *users*;  
 }  
}

Класс LoginWindow

**package** com.company;  
  
**import** javax.swing.\*;  
**import** javax.swing.border.EmptyBorder;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.util.ArrayList;  
  
  
**class** LoginWindow **extends** JFrame **implements** ActionListener {  
  
 *//Для того, чтобы впоследствии обращаться к содержимому текстовых полей, надо сделать их членами класса окна* JTextField **loginField**;  
 JPasswordField **passwordField**;  
 JButton **ok**;  
 JButton **cancel**;  
 PizzaSystem **pzsys**;  
  
  
 LoginWindow() {  
 **super**(**"Вход в систему"**);  
 **pzsys** = **new** PizzaSystem();  
 **this**.setVisible(**true**);  
 *// Настраиваем первую горизонтальную панель (для ввода логина)* Box box1 = Box.*createHorizontalBox*();  
 JLabel loginLabel = **new** JLabel(**"Логин: "**);  
 **loginField** = **new** JTextField(15);  
 box1.add(loginLabel);  
 box1.add(Box.*createHorizontalStrut*(6));  
 box1.add(**loginField**);  
  
 *// Настраиваем вторую горизонтальную панель (для ввода пароля)* Box box2 = Box.*createHorizontalBox*();  
 JLabel passwordLabel = **new** JLabel(**"Пароль:"**);  
 **passwordField** = **new** JPasswordField(15);  
 box2.add(passwordLabel);  
 box2.add(Box.*createHorizontalStrut*(6));  
 box2.add(**passwordField**);  
  
 *// Настраиваем третью горизонтальную панель (с кнопками)* Box box3 = Box.*createHorizontalBox*();  
 **ok** = **new** JButton(**"OK"**);  
 box3.add(Box.*createHorizontalGlue*());  
 box3.add(**ok**);  
 box3.add(Box.*createHorizontalStrut*(12));  
  
  
 *//Слушатели* **ok**.addActionListener(**this**);  
  
  
 *// Уточняем размеры компонентов loginLabel.setPreferredSize(passwordLabel.getPreferredSize());  
 // Размещаем три горизонтальные панели на одной вертикальной* Box mainBox = Box.*createVerticalBox*();  
 mainBox.setBorder(**new** EmptyBorder(12, 12, 12, 12));  
 mainBox.add(box1);  
 mainBox.add(Box.*createVerticalStrut*(12));  
 mainBox.add(box2);  
 mainBox.add(Box.*createVerticalStrut*(17));  
 mainBox.add(box3);  
 setContentPane(mainBox);  
 pack();  
 setResizable(**false**);  
 }  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 String message = **""**;  
 **if** (e.getSource() == **ok**) {  
 message = **""**;  
 **if** (**pzsys**.findUser(**loginField**.getText(), **passwordField**.getText())){  
 JOptionPane.*showMessageDialog*(**null**, **"Вы вошли в систему как: "**+ **pzsys**.getUser(), **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 mainWindow.*mapButton*.setEnabled(**true**);  
 mainWindow.*mainButton*.setEnabled(**true**);  
 mainWindow.*locationButton*.setEnabled(**true**);  
  
 mainWindow.*label*.setText(**"Здравствуйте, "**+ **pzsys**.getLogin());  
 mainWindow.*label1*.setText(**"Ваше местоположение: "**+**'\n'**+**pzsys**.getCurrentUser().getPlace());  
 **this**.dispose();  
 }  
 **else**{  
 JOptionPane.*showMessageDialog*(**null**, **"Неверный логин или пароль"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 }  
 **if** (e.getSource() == **cancel**) {  
  
 }  
  
 }  
}

Класс LocationWindow

**package** com.company;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
  
**public class** LocationWindow **extends** JFrame {  
 JButton **button** = **new** JButton(**"Локация: "**);  
 JLabel **label** = **new** JLabel();  
 JTextField **tf** = **new** JTextField(12);  
 LocationWindow() {  
 **super**(**"Выбор локации"**);  
 PizzaSystem pz = **new** PizzaSystem();  
 setSize(250,200);  
  
 JPanel pane = **new** JPanel();  
 add(pane);  
  
 **button**.setSize(50, 10);  
 **button**.setFont(**new** Font(**"Times New Roman"**, Font.***PLAIN***, 14));  
 **tf**.setSize(50, 10);  
  
 **label**.setText(**"Текущая локация курьера: "** + pz.getCurrentLocation());  
 Box box1 = Box.*createHorizontalBox*();  
 box1.add(**button**);  
 box1.add(Box.*createHorizontalGlue*());  
 box1.add(**tf**);  
 box1.add(Box.*createHorizontalStrut*(12));  
 Box box2 = Box.*createHorizontalBox*();  
 box2.add(**label**);  
 box2.add(Box.*createHorizontalGlue*());  
 box2.add(Box.*createHorizontalStrut*(12));  
 pane.add(box2);  
 pane.add(box1);  
 setVisible(**true**);  
 **button**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **if** (!**tf**.getText().equals(**""**)){  
 pz.setCurrentLocation(**tf**.getText());  
 **label**.setText(**"Текущая локация курьера: "** + pz.getCurrentLocation());  
 }  
 **else**{  
 JOptionPane.*showMessageDialog*(**null**, **"Поле не заполнено"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 }  
 });  
 }  
}

Класс UserSetWindow

**package** com.company;  
  
**import** javax.swing.\*;  
**import** javax.swing.border.EmptyBorder;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.IOException;  
  
**public class** UserSetWindow **extends** JFrame{  
 **private** PizzaSystem **pzsys**;  
 JTextField **t1** = **new** JTextField();  
 JTextField **t2** = **new** JTextField();  
 JTextField **t3** = **new** JTextField();  
 JTextField **t4** = **new** JTextField();  
 **public** UserSetWindow() {  
 **super**(**"Добавить пользователя"**);  
 **pzsys** = **new** PizzaSystem();  
  
 setSize(250,200);  
 Box box1 = Box.*createHorizontalBox*();  
 Box box2 = Box.*createHorizontalBox*();  
 Box box3 = Box.*createHorizontalBox*();  
 Box box4 = Box.*createHorizontalBox*();  
 Box box5 = Box.*createHorizontalBox*();  
 JPanel pane = (JPanel) getContentPane();  
 pane.setLayout(**null**);  
 JButton b1 = **new** JButton(**"Добавить"**);  
 JLabel label1 = **new** JLabel(**"Имя: "**);  
 JLabel label2 = **new** JLabel(**"Логин: "**);  
 JLabel label3 = **new** JLabel(**"Пароль: "**);  
 JLabel label4 = **new** JLabel(**"Место: "**);  
  
 box1.add(label1);  
 box1.add(**t1**);  
 box2.add(label2);  
 box2.add(**t2**);  
 box3.add(label3);  
 box3.add(**t3**);  
 box4.add(label4);  
 box4.add(**t4**);  
 box5.add(b1);  
 pane.add(box1);  
 pane.add(box2);  
 pane.add(box3);  
 pane.add(box4);  
 pane.add(box5);  
  
 Box mainBox = Box.*createVerticalBox*();  
 mainBox.setBorder(**new** EmptyBorder(12, 12, 12, 12));  
 mainBox.add(box1);  
 *//mainBox.add(Box.createVerticalStrut(5));* mainBox.add(box2);  
 *//mainBox.add(Box.createVerticalStrut(5));* mainBox.add(box3);  
 *//mainBox.add(Box.createVerticalStrut(5));* mainBox.add(box4);  
 *//mainBox.add(Box.createVerticalStrut(5));* mainBox.add(box5);  
  
 setContentPane(mainBox);  
 setVisible(**true**);  
 setResizable(**false**);  
 b1.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) { *//тут бы вхреначить проперку на существование пары логин-пароль, но мне влом* **if** (!**t1**.getText().equals(**""**) && !**t2**.getText().equals(**""**) && !**t3**.getText().equals(**""**) && !**t4**.getText().equals(**""**)) {  
 **pzsys**.addUser(**new** Client(**t1**.getText(), **t2**.getText(), **t3**.getText(), **t4**.getText()));  
 **try**{  
 **pzsys**.userSave();  
 dispose();  
 }**catch** (IOException ex){  
 System.***out***.println(**"Что-то пошло не так"** + ex.toString());  
 }  
  
 }  
 **else** {  
 JOptionPane.*showMessageDialog*(**null**, **"Не все поля заполнены"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 }  
 });  
 }  
  
}

Класс AdminWindow

**package** com.company;  
  
**import** javax.swing.\*;  
**import** javax.swing.table.AbstractTableModel;  
**import** javax.swing.table.DefaultTableModel;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.File;  
**import** java.util.ArrayList;  
  
**public class** AdminWindow **extends** JFrame {  
 PizzaSystem **pz**;  
 ArrayList<User> **users**;  
 JTable **table**;  
 AdminWindow(){  
 **super**(**"Таблица пользователей"**);  
 setVisible(**true**);  
 **try** {  
 **pz** = **new** PizzaSystem();  
 **pz**.userLoad();  
 }**catch** (Exception e){  
 System.***out***.println(e.toString());  
 JOptionPane.*showMessageDialog*(**null**, **"Дальше живут драконы"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 **this**.dispose();  
 }  
 **users** = **pz**.getUsers();  
  
 **table** = **new** JTable(**new** MyTableModel());  
 **table**.setPreferredScrollableViewportSize(**new** Dimension(950, 100));  
 JScrollPane jscrlp = **new** JScrollPane(**table**);  
 Box mainBox = Box.*createVerticalBox*();  
 mainBox.add(jscrlp);  
 setContentPane(mainBox);  
 pack();  
 }  
 **public class** MyTableModel **extends** AbstractTableModel {  
 @Override  
 **public int** getRowCount() {  
 **return users**.size();  
 }  
  
 @Override  
 **public** Object getValueAt(**int** r, **int** c) {  
 **switch** (c) {  
 **case** 0:  
 **return users**.get(r).getLogin();  
 **case** 1:  
 **return users**.get(r).getPassword();  
 **case** 2:  
 **return users**.get(r).getPlace();  
 **default**:  
 **return "этого здесь быть не должно"**;  
 }  
 }  
 @Override  
 **public** String getColumnName(**int** c) {  
 String result = **""**;  
 **switch** (c) {  
 **case** 0:  
 result = **"Логин"**;  
 **break**;  
 **case** 1:  
 result = **"Пароль"**;  
 **break**;  
 **case** 2:  
 result = **"Место"**;  
 **break**;  
  
 }  
 **return** result;  
 }  
 @Override  
 **public int** getColumnCount() {  
 **return** 3;  
 }  
 }  
}

Класс MapWindow

**package** com.company;  
  
**import** javax.swing.\*;  
**import** javax.swing.border.EmptyBorder;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
  
**public class** MapWindow **extends** JFrame {  
 JLabel **label**;  
 JButton **generateButton**;  
 JButton **pathButton**;  
 MapWindow(){  
 **super**(**""**);  
 PizzaSystem pz = **new** PizzaSystem();  
 **this**.setVisible(**true**);  
 *// Настраиваем первую горизонтальную панель* Box box1 = Box.*createHorizontalBox*();  
 **label** = **new** JLabel(**"Вы пока не сгенерировали путь"**);  
 box1.add(**label**);  
 box1.add(Box.*createHorizontalStrut*(6));  
  
 *// Настраиваем вторую горизонтальную панель* Box box2 = Box.*createHorizontalBox*();  
 **generateButton** = **new** JButton(**"Сгенерировать маршрут"**);  
 **generateButton**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **if** (pz.getCurrentLocation() == **null**){  
 JOptionPane.*showMessageDialog*(**null**, **"Вы не уточнили положение курьера"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 **if** (pz.getCurrentUser().getPlace() == **null**){  
 JOptionPane.*showMessageDialog*(**null**, **"У вас нет местоположения"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 **if** (!pz.getMap().getPoints().contains(pz.getCurrentLocation())){  
 JOptionPane.*showMessageDialog*(**null**, **"вы ввели место курьера, отсутствующее на карте"**, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 **else** {  
 **label**.setText(pz.generate(pz.getCurrentUser().getPlace(), pz.getCurrentLocation()));  
 }  
 }**catch** (Exception ex){  
 System.***out***.println(ex.toString());  
 String message = **"Что-то пошло не так, возможно, маршрута несуществует"**;  
 JOptionPane.*showMessageDialog*(**null**, message, **""**, JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 }  
 });  
 box2.add(**generateButton**);  
 box2.add(Box.*createHorizontalStrut*(6));  
 **pathButton** = **new** JButton(**"Настроить пути"**);  
 **pathButton**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 PathTableWindow tt = **new** PathTableWindow();  
 }  
 });  
 box2.add(**pathButton**);  
  
 *// Уточняем размеры компонентов  
 // Размещаем три горизонтальные панели на одной вертикальной* Box mainBox = Box.*createVerticalBox*();  
 mainBox.setBorder(**new** EmptyBorder(12, 12, 12, 12));  
 mainBox.add(box1);  
 mainBox.add(Box.*createVerticalStrut*(12));  
 mainBox.add(box2);  
 setContentPane(mainBox);  
 pack();  
 setResizable(**false**);  
  
 }  
}

Класс mainWindow

**package** com.company;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
  
**public class** mainWindow **extends** JFrame{  
 **static** JButton *userButton* = **new** JButton(**"Зарегистрироваться"**);  
 **static** JButton *mainButton* = **new** JButton(**"Посмотреть зарегестрированных пользователей"**);  
 **static** JButton *loginButton* = **new** JButton(**"Войти в систему"**);  
 **static** JButton *mapButton* = **new** JButton(**"Настроить карту"**);  
 **static** JButton *locationButton* = **new** JButton(**"Местоположение курьера"**);  
 **static** JLabel *label* = **new** JLabel(**"Вы не авторизированы"**);  
 **static** JLabel *label1* = **new** JLabel(**""**);  
 **static** JFrame *frame* = **new** JFrame();  
 **public** mainWindow(){  
 }  
  
  
  
 **public static void** main(String[] args) **throws** IOException, ClassNotFoundException {  
 PizzaSystem ps = **new** PizzaSystem();  
 **try** {  
 ps.userLoad();  
 }**catch** (IOException | ClassNotFoundException e){  
 System.***out***.println(**"Что-то пошло не так "** + e.toString());  
 }  
  
 *frame*.setTitle(**"Главное окно"**);  
 *frame*.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 *frame*.setBounds(100,100,280,500);  
  
 *frame*.setVisible(**true**);  
  
 Container container = *frame*.getContentPane();  
 container.setLayout(**null**);  
 *label*.setLocation(10, 0);  
 *label*.setSize(250,20);  
 *label1*.setLocation(10, 15);  
 *label1*.setSize(250,20);  
 container.add(*label*);  
 container.add(*label1*);  
 *loginButton*.setSize(250,30);  
 *loginButton*.setLocation(10,40);  
 *userButton*.setSize(250,30);  
 *userButton*.setLocation(10,80);  
 *mainButton*.setEnabled(**false**);  
 *mainButton*.setSize(250,30);  
 *mainButton*.setLocation(10,120);  
 *mapButton*.setEnabled(**false**);  
 *mapButton*.setSize(250,30);  
 *mapButton*.setLocation(10,160);  
 *locationButton*.setEnabled(**false**);  
 *locationButton*.setSize(250,30);  
 *locationButton*.setLocation(10,200);  
 container.add(*loginButton*);  
 container.add(*userButton*);  
 container.add(*mainButton*);  
 container.add(*mapButton*);  
 container.add(*locationButton*);  
 container.setFont(**new** Font(**"Times New Roman"**, Font.***PLAIN***, 14));  
  
 *loginButton*.addActionListener(**new** UserButtonListener());  
 *userButton*.addActionListener(**new** LoginButtonListener());  
 *locationButton*.addActionListener(**new** LocationButtonListener());  
 *mapButton*.addActionListener(**new** MapButtonListener());  
 *mainButton*.addActionListener(**new** AdminButtonListener());  
  
 }  
 **static class** UserButtonListener **implements** ActionListener{  
 **public void** actionPerformed(ActionEvent e){  
 LoginWindow loginWindow = **new** LoginWindow();  
 }  
 }  
 **static class** LoginButtonListener **implements** ActionListener{  
 **public void** actionPerformed(ActionEvent e){  
 UserSetWindow l = **new** UserSetWindow();  
 }  
 }  
 **static class** LocationButtonListener **implements** ActionListener{  
 **public void** actionPerformed(ActionEvent e){  
 LocationWindow lw = **new** LocationWindow();  
 }  
 }  
 **static class** MapButtonListener **implements** ActionListener{  
 **public void** actionPerformed(ActionEvent e){  
 MapWindow t = **new** MapWindow();  
 }  
 }  
 **static class** AdminButtonListener **implements** ActionListener{  
 **public void** actionPerformed(ActionEvent e){  
 AdminWindow aw = **new** AdminWindow();  
 }  
 }  
}