ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title			
Submission date	24/06/2022	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Bùi Hương Linh	**Student ID**	GBH200662
Class	GCH1002	**Assessor name**	Đinh Đức Mạnh
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	Linh

Grading grid

Grade (0-10)

❒Summative Feedback:		❒Resubmission Feedback:
Grade:	**Assessor Signature:**	**Date:**
IV Signature:		

Table of Contents

I. Introduction

Currently, the use of hotels for relaxation as well as entertainment is increasingly popular. This leads to manual customer management which makes it difficult, delayed and out of control of the right quantity. Therefore, I decided to create an application to support customer management when booking in java language. This application helps me to know the exact customer information and control the price as well as the number of rooms, the date that the customer wants to stay.

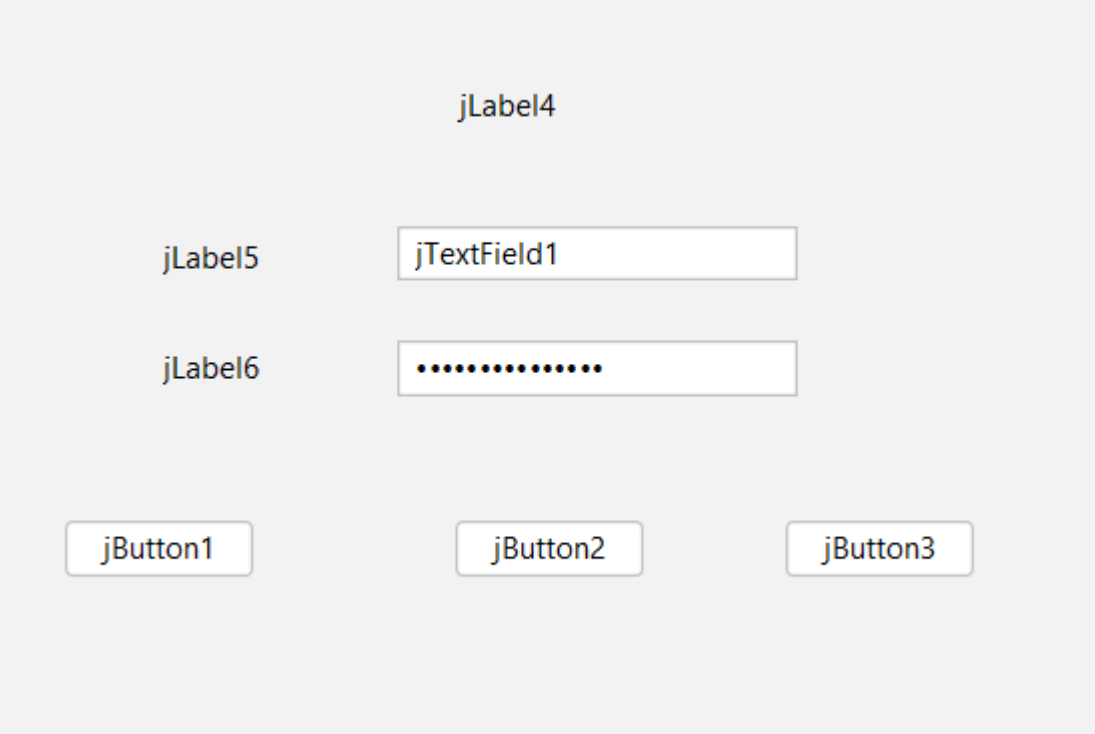
II. Requirement

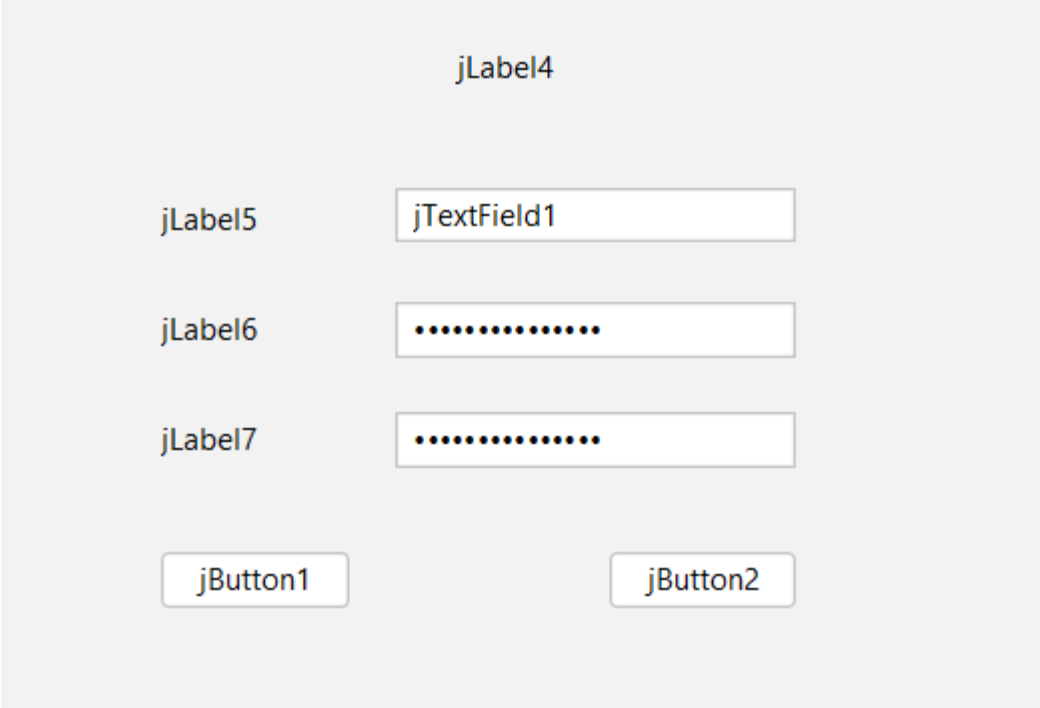
The java program that I created helped me solve some of the following problems:

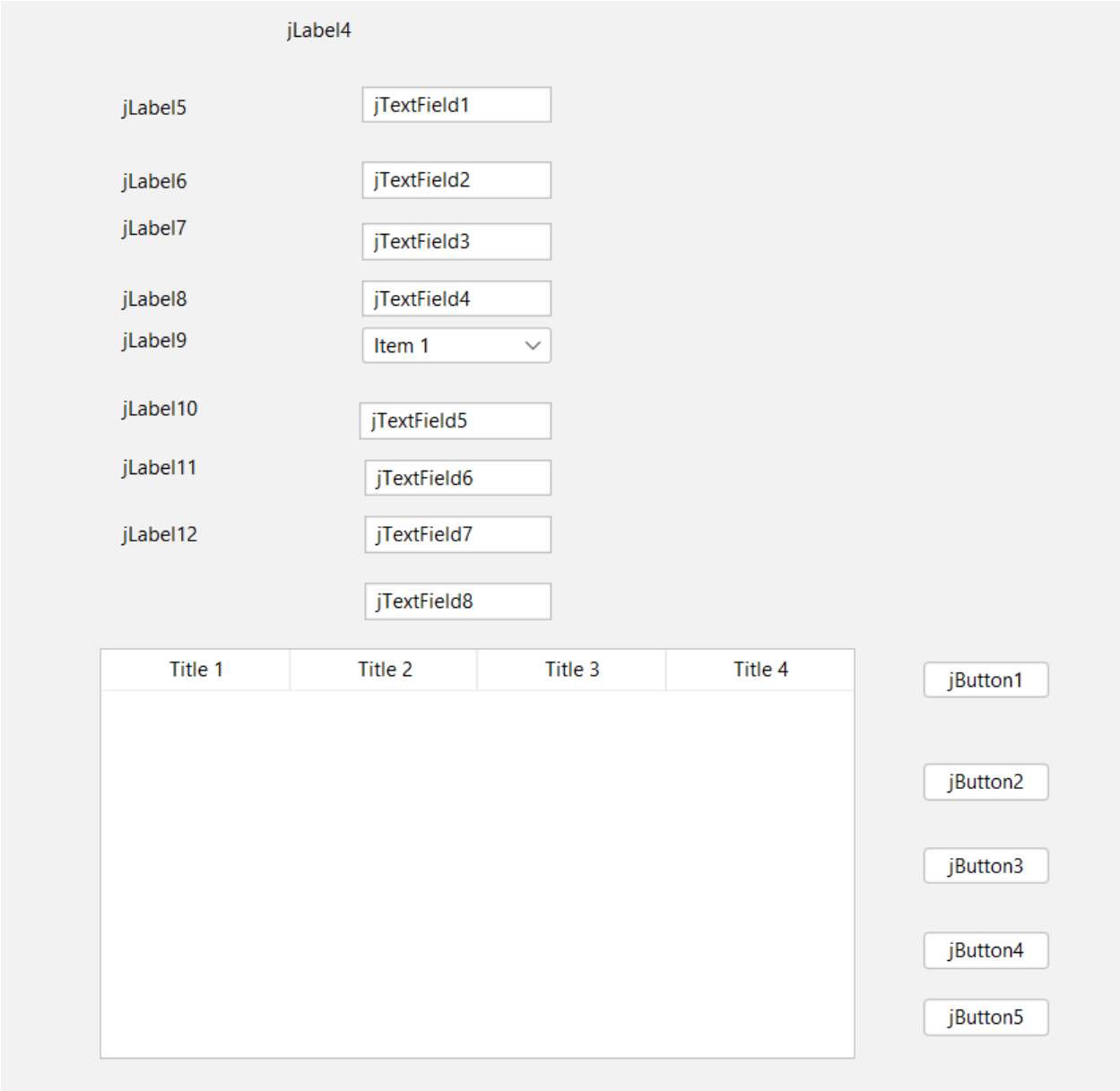
- I can insert, edit, delete, update, insert new and search them easily and understandably.
- I can find the customer information I need through the search function.
- Register a new user for the system.
- Save file to computer.
- Read file from computer.

III. UI design

The basic user interface of the program will have labels, text fields and combo boxes for the user to enter the appropriate data. In addition, the user can interact with the entered data through buttons. Finally, display a table of the entries' results.



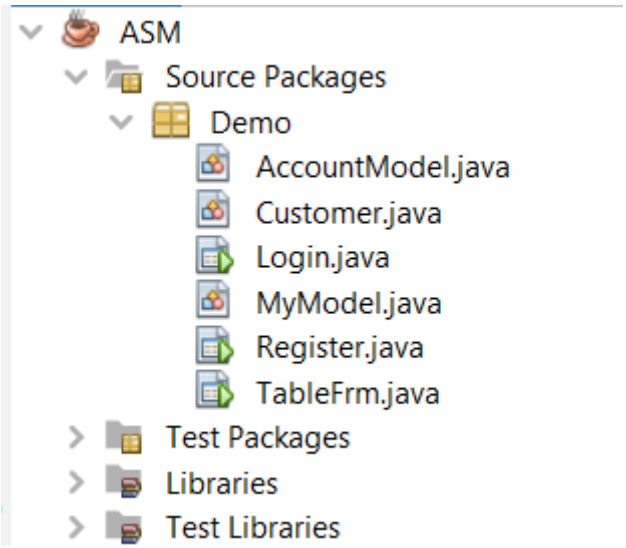

jLabel4
jLabel5
jTextField1
jLabel6
jLabel7
jButton1
jButton2



IV. Implementation

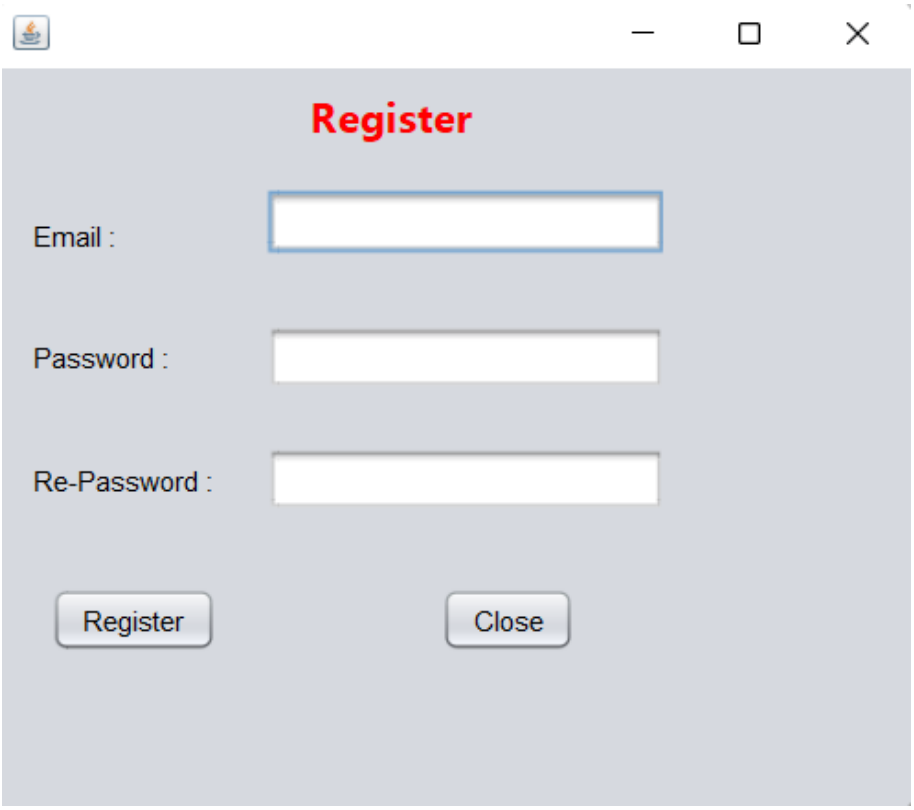
1. Explain program structure

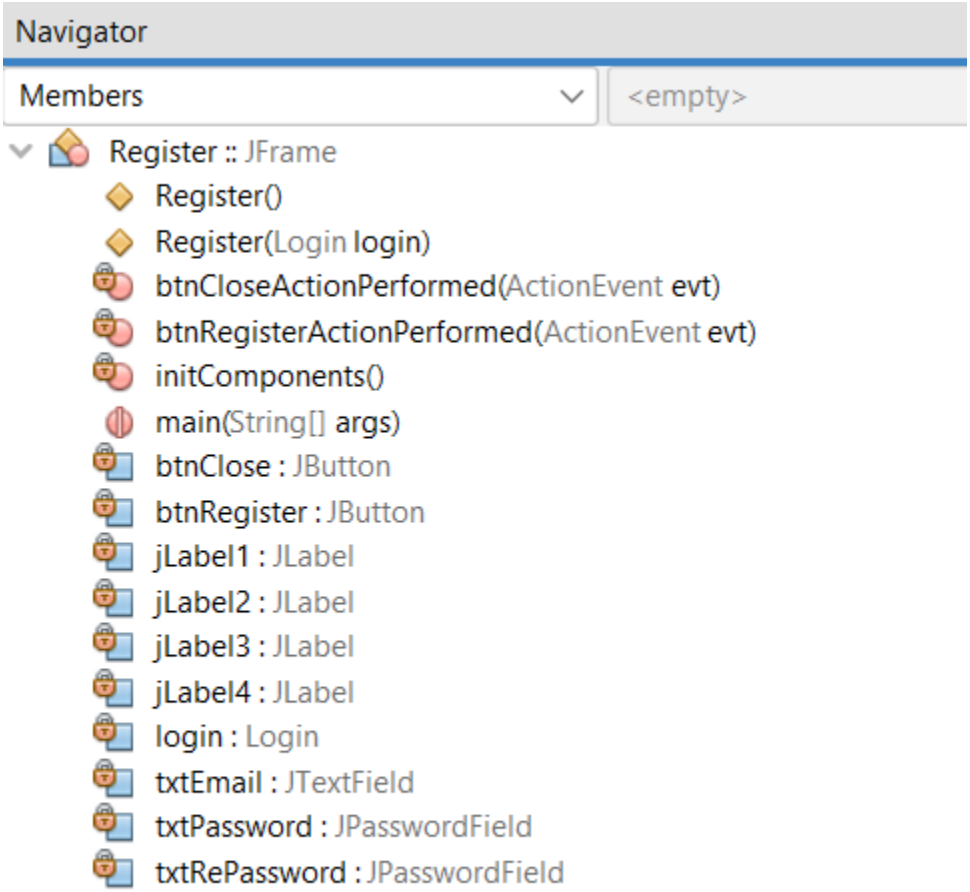
   I create a java application, which consists of 6 main parts: 3 classes named Customer.java, MyModel.java and AccountModel.java and 3 Jframes named TableFrm.java, Login.java and Register.java.



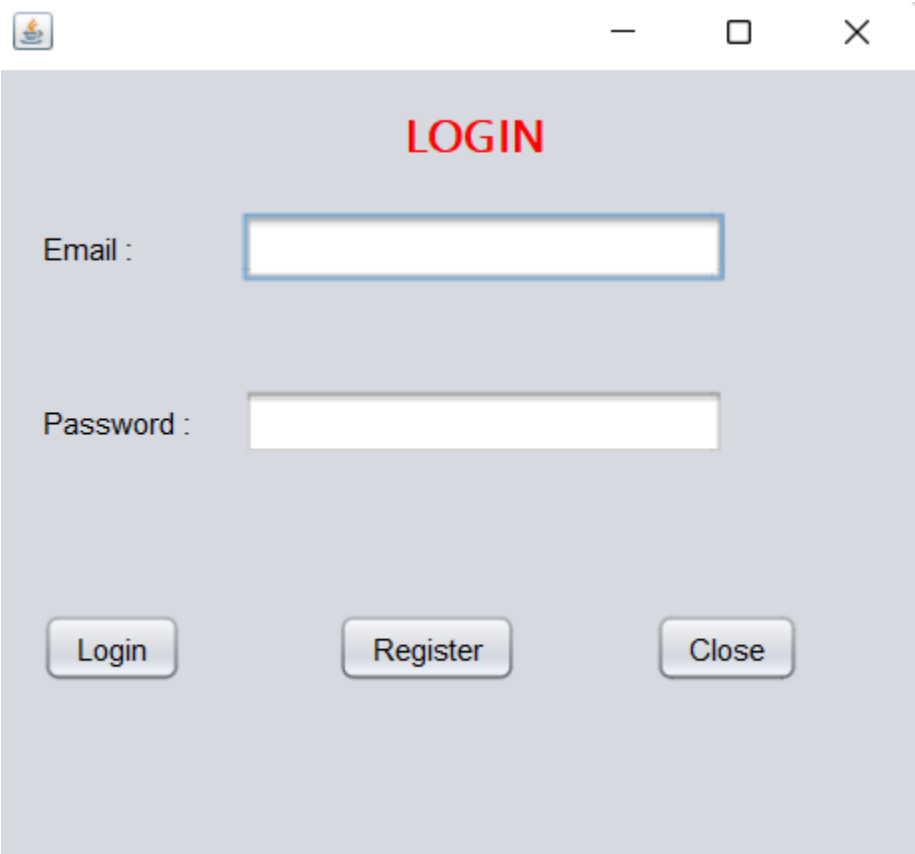
Then the class I created is used to declare customer variables, mymodel, accountmodel with methods. Then I will use them to populate the Jframe I created. I'll develop functions for my software using it.
I based my program interface design on the wireframe I had already constructed, giving each piece a variable name (button, text field, label, table, combo box,...)

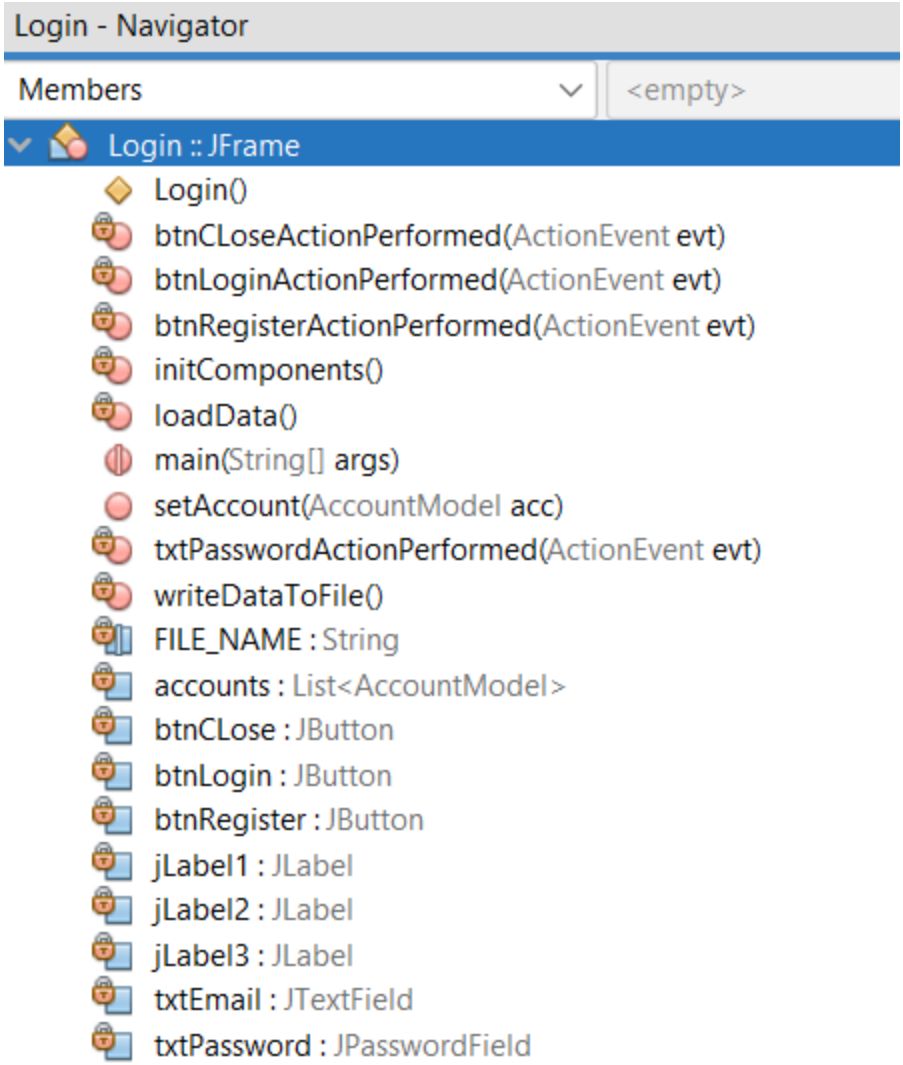
1.1. Register.java



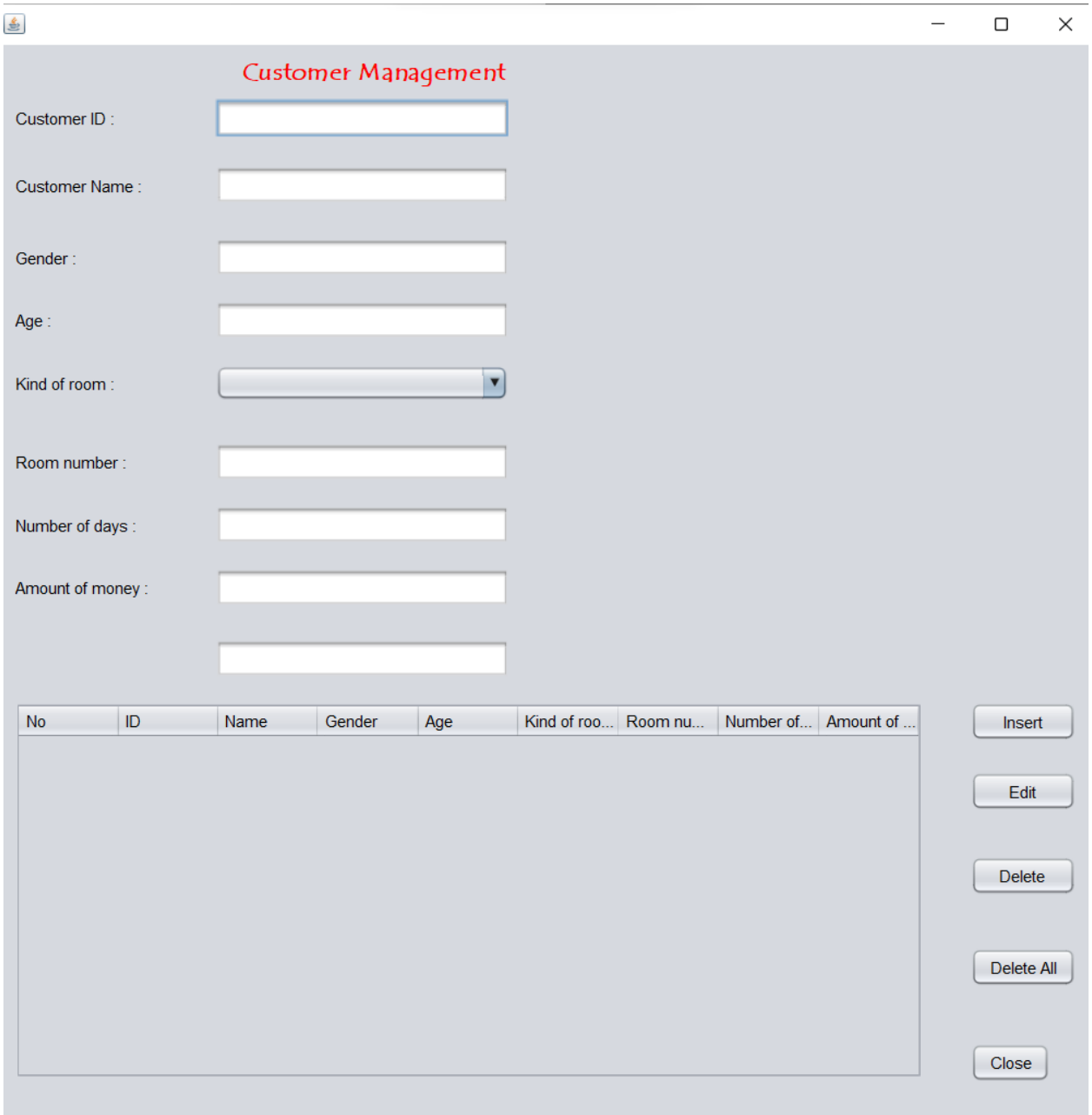

Navigator
Members
<empty>
Register :: JFrame
Register()
Register(Login login)
btnCloseActionPerformed(ActionEvent evt)
btnRegisterActionPerformed(ActionEvent evt)
initComponents()
main(String[] args)
btnClose : JButton
btnRegister : JButton
jLabel1 : JLabel
jLabel2 : JLabel
jLabel3 : JLabel
jLabel4 : JLabel
login : Login
txtEmail : JTextField
txtPassword : JPasswordField
txtRePassword : JPasswordField

1.2.Login.java




Login - Navigator
Members
<empty>
Login :: JFrame
Login()
btnCLoseActionPerformed(ActionEvent evt)
btnLoginActionPerformed(ActionEvent evt)
btnRegisterActionPerformed(ActionEvent evt)
initComponents()
loadData()
main(String[] args)
setAccount(AccountModel acc)
txtPasswordActionPerformed(ActionEvent evt)
writeDataToFile()
FILE_NAME : String
accounts : List<AccountModel>
btnCLose : JButton
btnLogin : JButton
btnRegister : JButton
jLabel1 : JLabel
jLabel2 : JLabel
jLabel3 : JLabel
txtEmail : JTextField
txtPassword : JPasswordField

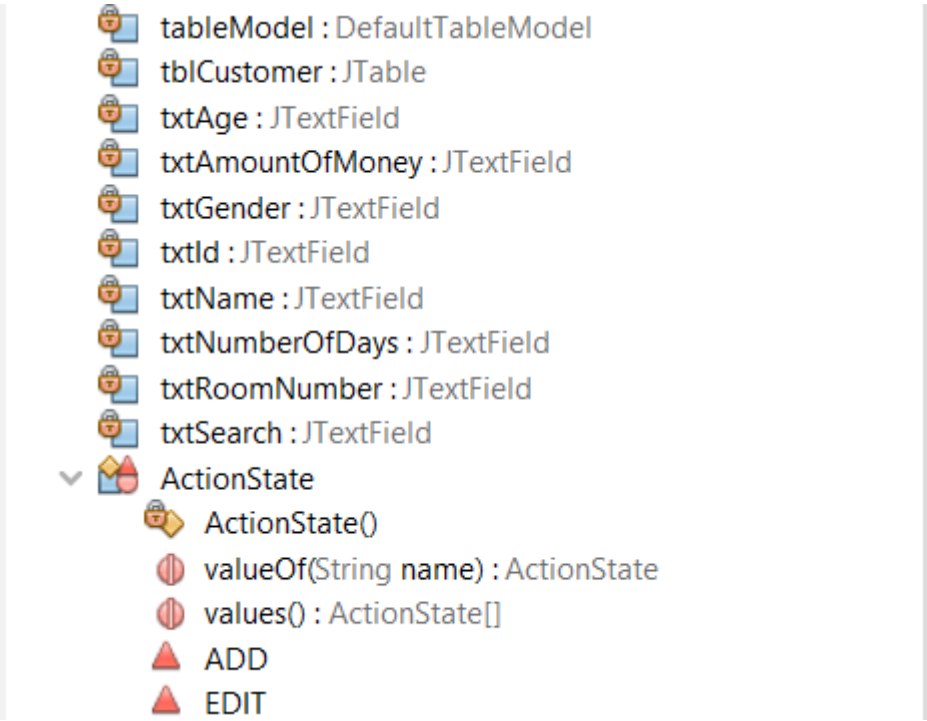
1.3. TableFrm.java



TableFrm.java - Navigator

Members | <empty>

- TableFrm :: JFrame
  - TableFrm(AccountModel acc)
  - TableFrm()
  - btlClearActionPerformed(ActionEvent evt)
  - btnAddActionPerformed(ActionEvent evt)
  - btnCloseActionPerformed(ActionEvent evt)
  - btnDelActionPerformed(ActionEvent evt)
  - btnEditActionPerformed(ActionEvent evt)
  - initComponents()
  - main(String[] args)
  - showCustomers(Customer customer)
  - txtSearchActionPerformed(ActionEvent evt)
  - updateTable(Customer customer)
  - actionState : ActionState
  - btlClear : JButton
  - btnAdd : JButton
  - btnClose : JButton
  - btnDel : JButton
  - btnEdit : JButton
  - buttonGroup1 : ButtonGroup
  - cbKindOfRoom : JComboBox<String>
  - customers : List<Customer>
  - editedIndex : int
  - index : int
  - jLabel1 : JLabel
  - jLabel2 : JLabel
  - jLabel3 : JLabel
  - jLabel4 : JLabel
  - jLabel5 : JLabel
  - jLabel6 : JLabel
  - jLabel7 : JLabel
  - jLabel8 : JLabel
  - jLabel9 : JLabel
  - jScrollPane1 : JScrollPane
  - tableModel : DefaultTableModel



2. Explain classes

2.1. Class Customer (Customer.java)

- Import the necessary libraries to use the defined variables.
- In this section, I have declared all local variables with respective types (string, integer, float...) as private because when declaring private can only be accessed within the declared class itself there.

```
package Demo;

import java.util.Objects;

public class Customer {

    private String name;
    private String id;
    private String kindofroom;
    private String age;
    private String gender;
    private String room;
    private String day;
    private String money;
```

- I'll initialize the methods first. I use the aforementioned declarations to initialize the objects. I was able to call other equivalent methods to initialize the logic inside the objects as well as initialize all the values for the internal properties thanks to this.

```
public Customer() {
}

Customer(String name, String id, String kindofroom, String room, String day, String age, String money, String gender) {
    this.name = name;
    this.id = id;
    this.kindofroom = kindofroom;
    this.age = age;
    this.gender = gender;
    this.room = room;
    this.day = day;
    this.money = money;
}
```

- I applied Java encapsulation using getter() and setter() methods, once accessibility was defined for the private properties I created above, and those properties were given view and edit.

```java
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getKindofroom() {
        return kindofroom;
    }

    public void setKindofroom(String kindofroom) {
        this.kindofroom = kindofroom;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
```

```java
    public String getRoom() {
        return room;
    }

    public void setRoom(String room) {
        this.room = room;
    }

    public String getDay() {
        return day;
    }

    public void setDay(String day) {
        this.day = day;
    }

    public String getMoney() {
        return money;
    }

    public void setMoney(String money) {
        this.money = money;
    }
```

- To this class, I'm using inheritance in Java. I can use the hashCode() function to get the hashCode value of the aforementioned initialized components. Additionally, I compare two objects semantically using the equals() method.

```java
    @Override
    public int hashCode() {
        int hash = 7;
        hash = 97 * hash + Objects.hashCode(this.name);
        hash = 97 * hash + Objects.hashCode(this.id);
        hash = 97 * hash + Objects.hashCode(this.kindofroom);
        hash = 97 * hash + Objects.hashCode(this.age);
        hash = 97 * hash + Objects.hashCode(this.gender);
        hash = 97 * hash + Objects.hashCode(this.room);
        hash = 97 * hash + Objects.hashCode(this.day);
        hash = 97 * hash + Objects.hashCode(this.money);
        return hash;
    }
```

```java
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Customer other = (Customer) obj;
    if (!Objects.equals(this.name, other.name)) {
        return false;
    }
    if (!Objects.equals(this.id, other.id)) {
        return false;
    }
    if (!Objects.equals(this.kindofroom, other.kindofroom)) {
        return false;
    }
    if (!Objects.equals(this.age, other.age)) {
        return false;
    }
    if (!Objects.equals(this.gender, other.gender)) {
        return false;
    }
    if (!Objects.equals(this.room, other.room)) {
        return false;
    }
    if (!Objects.equals(this.day, other.day)) {
        return false;
    }
    return Objects.equals(this.money, other.money);
}
```

2.2. Class AccountModel(Accountmodel.java)

- Import the necessary libraries to use the defined variables.
- In this section, I have declared all local variables with respective types

(string, integer, float...) as private because when declaring private can only be accessed within the declared class itself there.

- Similar to the explanation of the customer.java .

```java
package Demo;


import java.util.Objects;

public class AccountModel {
    private String Email;
    private String Password;


    public AccountModel() {
    }

    AccountModel(String Email, String Password) {
        this.Email = Email;
        this.Password = Password;
    }

    public String getEmail() {
        return Email;
    }

    public void setEmail(String email) {
        this.Email = email;
    }

    public String getPassword() {
        return Password;
    }

    public void setPassword(String password) {
        this.Password = password;
    }
```

```java
    @Override
    public int hashCode() {
        int hash = 7;
        hash = 13 * hash + Objects.hashCode(this.Email);
        hash = 13 * hash + Objects.hashCode(this.Password);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final AccountModel other = (AccountModel) obj;
        if (!Objects.equals(this.Email, other.Email)) {
            return false;
        }
        return Objects.equals(this.Password, other.Password);
    }

}
```

2.3. Class MyModel(MyModel.java)

- Similar to the explanation of the customer.java .
- Import all necessary libraries. Create a class mymodel that inherits comboboxmodel of type string.

```java
package Demo;

import java.util.ArrayList;
import java.util.List;
import javax.swing.ComboBoxModel;
import javax.swing.event.ListDataListener;

public class MyModel implements ComboBoxModel<String> {

    private List<String> kindofroom;
    private Object seleObject;

    public MyModel() {
        kindofroom = new ArrayList<>();
        kindofroom.add("Affordable");
        kindofroom.add("Vip");
        kindofroom.add("President");

    }
```

```java
    @Override
    public void setSelectedItem(Object anItem) {
        seleObject = anItem;
    }

    @Override
    public Object getSelectedItem() {
        return seleObject;
    }

    @Override
    public int getSize() {
        return kindofroom.size();
    }

    @Override
    public String getElementAt(int index) {
        return kindofroom.get(index);
    }

    @Override
    public void addListDataListener(ListDataListener l) {
    }

    @Override
    public void removeListDataListener(ListDataListener l) {
    }
}
```

3. Explain important algorithms

3.1. Register

- Import all necessary libraries

```java
package Demo;

import javax.swing.JOptionPane;
```

```java
public class Register extends javax.swing.JFrame {

    private Login login;

    public Register() {
        initComponents();
    }

    public Register(Login login) {
        this();
        this.login = login;
    }
```

- Get data from users; if entered correctly, it will display the message "Account creation successful," but if incorrectly entered or left blank, the error message "All cases cannot be blank" will be shown. The information is then put into a list using setAccount.

```java
private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    var Email = txtEmail.getText();// gasn vaof object email
    var Password = new String(txtPassword.getPassword());
    var rePassword = new String(txtRePassword.getPassword());
    if (!Email.isEmpty() && !Password.isEmpty() && !rePassword.isEmpty()) {
        if (Password.compareTo(rePassword) == 0 ) {
            AccountModel acc = new AccountModel(Email, Password );
            login.setAccount(acc);
            JOptionPane.showMessageDialog(rootPane, "Tạo tài khoản thành công!");
            this.dispose();
        } else {
            JOptionPane.showMessageDialog(rootPane, "Tất cả các trường không được để trống!");
        }
    } else {
        JOptionPane.showMessageDialog(rootPane, "Tất cả các trường không được để trống!");
    }
}
```

- Exit
  Exit system

```java
private void btnCloseActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

3.2. Login

- Import all necessary libraries.

```java
package Demo;

import java.util.List;
import java.util.ArrayList;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.swing.JOptionPane;
```

- Create a list containing the ‘AccountModel’ assigned to ‘accounts’. Create a file-name = "123.txt' of type string.

```java
public class Login extends javax.swing.JFrame {

    private List<AccountModel> accounts;
    private static final String FILE_NAME = "123.txt";

    public Login() {
        initComponents();
        loadData();
    }
```

- Get the value from the UI variable and assign it to an intermediate variable. Create an 'AccountModel' containing the email and password assigned to an intermediate variable. If 'account ! = null' and 'account.contains(acc)' will bring up the table 'TableFrm'. Otherwise, an error 'Email or password is incorrect' will be displayed.

```java
private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    var email = txtEmail.getText(); //getText gán vào object email
    var password = new String(txtPassword.getPassword());
    AccountModel acc = new AccountModel(email, password);
    if (accounts != null && accounts.contains(acc)) {
        TableFrm table = new TableFrm(acc);
        table.setVisible(true);
        this.dispose();
    } else {
        JOptionPane.showMessageDialog(rootPane, "Email hoac mat khau ko dung!");
    }
}
```

- If the customer does not have an account, when logging in, the customer will be taken to the registration interface before logging into the account.

```
private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    Register register = new Register(this);// tạo register
    register.setVisible(true);
}
```

- Create a file containing the 'File_Name' created above for the variable 'fis'. Create object of 'fi's for variable 'ois'. Read an object in the list that has 'AccountModel' assigned to 'accounts'. There are 3 cases: FileNotFoundException, IOException, ClassNotFoundException.

```
private void loadData() {
    try ( FileInputStream fis = new FileInputStream(FILE_NAME);//
          ObjectInputStream ois = new ObjectInputStream(fis)) {//
        accounts = (List<AccountModel>) ois.readObject();// lấy d
    } catch (FileNotFoundException ex) {// k tìm thấy file
        ex.printStackTrace();
    } catch (IOException ex) { // viết nhầm
        ex.printStackTrace();
    } catch (ClassNotFoundException ex) {// k tìm thấy class
        ex.printStackTrace();
    }
}
```

- Create file output stream containing 'File_Name' create above for variable 'fos'. Creates an ObjectOutputStream that writes to the specified 'fos' assigned to 'oos'. Write an object to the stream. 2 cases appear: FileNotFoundException, IOException.

```
private void writeDataToFile() {
    try (FileOutputStream fos = new FileOutputStream(FILE_NAME);// x
            ObjectOutputStream oos = new ObjectOutputStream(fos)) {
        oos.writeObject(accounts);// viết data
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

- If 'accounts == null' create an 'ArrayList' assigned to 'accounts'. Then add the object to 'accounts'. Finally load the data into the file.

```
    public void setAccount(AccountModel acc){
        if(accounts == null){ // nếu accounts
            accounts = new ArrayList<>();// tạ
        }
        accounts.add(acc);// add acc vào accou
        writeDataToFile();// load giữu liệu và
    }
```

- Exit
  Exit system

```
    private void btnCLoseActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }
```

3.3. TableFrm

- I built my program on JFrame. I've first imported the libraries I'll be using in my program.

```
package Demo;

import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.RowFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
```

- I constructed the AccountModel.java class and have declared properties, a new List named customer that is connected from it, a memory variable called model to use. Two integer variables, index and editIndex, are utilized to show and interact with information in the information display function.
-

```java
public class TableFrm extends javax.swing.JFrame {

    private List<Customer> customers;

    private DefaultTableModel tableModel;
    private int index;
    private int editedIndex;//row index

    TableFrm(AccountModel acc) {
        this();
    }

    private enum ActionState {
        ADD, EDIT
    }
    private ActionState actionState; // edit/add state

    public TableFrm() {
        initComponents();
        setLocationRelativeTo(null);
        customers = new ArrayList<>();
        tableModel = (DefaultTableModel) tblCustomer.getModel();
        index = 1;
        editedIndex = -1;
        actionState = ActionState.ADD;
    }
```

- Get value from UI assign to intermediate variable of type var. If id, name, gender, age,... are not empty and 'cbKindOfRoom' is not null. Get 'selectedItem' of type String to an intermediate variable of type var. Create 'Customer' containing id, name, room, day, gender, ... assign it to a variable of type var. If add is correct, it will display " Insert oki '. Otherwise, it will display the message 'The inputs cannot be blank'

```java
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    var id = txtId.getText(); // lấy giá trị ob gán vào biến trung gian
    var name = txtName.getText();
    var room = txtRoomNumber.getText();
    var age = txtAge.getText();
    var day = txtNumberOfDays.getText();
    var money = txtAmountOfMoney.getText();
    var gender = txtGender.getText();

    if (!id.isEmpty() && !name.isEmpty() && !room.isEmpty() && !age.isEmpty() && !gender.isEmpty()
        && !day.isEmpty() && !money.isEmpty()) {// nếu id, name... không trống và cbkindogroom k nu
        if (cbKindOfRoom.getSelectedItem() != null) {
            var kindofroom = cbKindOfRoom.getSelectedItem().toString();// gán selectedItem kiểu str
            var customer = new Customer(name, id, kindofroom, room, day, age, money, gender);// tạo
            if (actionState == ActionState.ADD) {

                if (customers.contains(customer)) {
                    JOptionPane.showMessageDialog(rootPane,
                            "Customer ID \"" + id + "\"exits!");
                } else {
                    customers.add(customer);//

                    showCustomers(customer);
                    JOptionPane.showMessageDialog(rootPane,
                            "Insert OK!");
                }

            } else if (actionState == ActionState.EDIT) {
                updateTable(customer);
                actionState = ActionState.ADD;
                btnAdd.setText("Insert new");
            }

            txtId.setText("");
            txtName.setText("");
            txtRoomNumber.setText("");
            txtNumberOfDays.setText("");
            txtGender.setText("");
            txtAge.setText("");
            txtAmountOfMoney.setText("");

        } else {
            JOptionPane.showMessageDialog(rootPane,
                    "Major combobox cannot be blank!");
        }
    } else {
        JOptionPane.showMessageDialog(rootPane,
                "The inputs cannot be blank!");
    }
}
```

- If customers.size > 0 then assign a selection of the table. Get the value in the table assign to 'editedindex'. If selected line is different from '-1' then get data from 'customers' assign it to 'customer' in type var and reset. done we use for loop vs 'i=0; i<comboBoSize; i++'. if 2 ob =0 then will set the selected item in 'cbKindOfRoom' and 'add' will change to 'update'. Otherwise, the error message 'Please choose a row' will appear. Else the error message 'The inputs cannot the blank' appears.

```
private void btnEditActionPerformed(java.awt.event.ActionEvent evt) {
    if (customers.size() > 0) { // cus,sz > 0 thì gán 1 luuaj chọnc của bảng
        editedIndex = tblCustomer.getSelectedRow();// laasy gtri trong bangr gasn cho edu=
        if (editedIndex != -1) { // nếu dòng đc đc chọn khác -1
            var customer = customers.get(editedIndex);// lay du lieu tuw cuss vao cus
            txtId.setText(customer.getId()); // thiết lập lai
            txtName.setText(customer.getName());
            txtRoomNumber.setText(customer.getRoom());
            txtGender.setText(customer.getGender());
            txtNumberOfDays.setText(customer.getDay());
            txtAmountOfMoney.setText(customer.getMoney());
            txtAge.setText(customer.getAge());

            int comboBoxSize = cbKindOfRoom.getItemCount();// lay gtri trong cbkind gan ch
            for (int i = 0; i < comboBoxSize; i++) {
                if (customer.getKindofroom().compareTo(cbKindOfRoom.getItemAt(i)) == 0) {
                    cbKindOfRoom.setSelectedIndex(i);
                    break;
                }
            }
            btnAdd.setText("Update");
            actionState = ActionState.EDIT;

        } else {
            JOptionPane.showMessageDialog(rootPane,
                    "Please choose a row!");
        }

    } else {
        JOptionPane.showMessageDialog(rootPane,
                "The inputs cannot be blank!");
    }
}
```

- Set 'edittedIndex and customer' for 'customers'. done, delete the row that we have selected in the table. Create an object and assign it to a variable of type var. data in 'customer' will pass in attributes such as: id, name, age, room, day.... Finish, inserting all the properties that the newly created variable has received into the 'tableModel' table. finally update the data in the table 'tableModel'.

```
private void updateTable(Customer customer) {
    customers.set(editedIndex, customer);// thiết lập edittedIndex và cús cho cus thêm s
    tableModel.removeRow(editedIndex); // xóa dòng mà ta đã chọn ở bảng
    var row = new Object[]{ // tạo ob và gán cho biến trung gian
        editedIndex + 1, customer.getId(), customer.getName(), customer.getKindofroom(), customer.getAge(),
                 customer.getGender(), customer.getDay(), customer.getMoney(), customer.getRoom() };
                 // dữ liệu trong customer sẽ truyền vào các thuộc tnhs như id, name...
    tableModel.insertRow(editedIndex, row);// insert all các thuốc tính mà row đã nhận
    tableModel.fireTableDataChanged(); // cập nhập lại dữ liệu trong bảng
}
```

- When the user selects a row in the table, if the selected row is different from '-1' then delete appears the message 'Do you want to delete' and when press yes, the message 'Delete ok' will appear. otherwise, the error message 'Please choose a row to delete' appears. otherwise the error message appears 'Empty list'.

```
private void btnDelActionPerformed(java.awt.event.ActionEvent evt) {
    if (customers.size() > 0) {
        int selectedIndex = tblCustomer.getSelectedRow();
        if (selectedIndex != -1) {
            int choice = JOptionPane.showConfirmDialog(rootPane, "Do you want to delete this row?");
            if (choice == JOptionPane.YES_OPTION) {
                customers.remove(selectedIndex);
                tableModel.removeRow(selectedIndex);
                tableModel.fireTableDataChanged();
                JOptionPane.showMessageDialog(rootPane, "Delete ok!");
            } else {
                JOptionPane.showMessageDialog(rootPane, "Please choose a row to delete!");
            }
        } else {
            JOptionPane.showMessageDialog(rootPane, "Empty list!");
        }
    }
}
```

- When the user wants to delete all, the message 'Do you want to delete all rows' appears and when pressing yes, the message 'Delete ok' will appear. Else the error message appears 'Empty list'.

```
private void btlClearActionPerformed(java.awt.event.ActionEvent evt) {
    if (customers.size() > 0) {
        int choice = JOptionPane.showConfirmDialog(rootPane, "Do you want to delete all rows?");
        if (choice == JOptionPane.YES_OPTION) {
            for (int i = customers.size() - 1; i >= 0; i--) {
                tableModel.removeRow(i);
            }
            customers.clear();

            tableModel.fireTableDataChanged();
            JOptionPane.showMessageDialog(rootPane, "Delete ok!");
        } else {
            JOptionPane.showMessageDialog(rootPane, "Emty list !");
        }
    }
}
```

- At the Search function I have initialized the event for it. get Model from table assign to variable. Get value from UI variable convert from chuooix to lowercase assign to variable 'search' of type string. create a 'TableRowSorter' contains 'table' assigned to 'trs'. Then set up the table.

```
private void txtSearchActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultTableModel table = (DefaultTableModel) tblCustomer.getModel();
    String search = txtSearch.getText().toLowerCase(); // toLowerCase: chu
    TableRowSorter<DefaultTableModel> trs = new TableRowSorter<>(table);
    tblCustomer.setRowSorter(trs);
    trs.setRowFilter(RowFilter.regexFilter(search));
}
```

- Create an object and assign it to a variable of type var. data in 'customer' will pass in attributes such as: id, name, age, room, day.... Finish, inserting all the properties that the newly created variable has received into the 'tableModel' table. finally update the data in the table 'tableModel'.

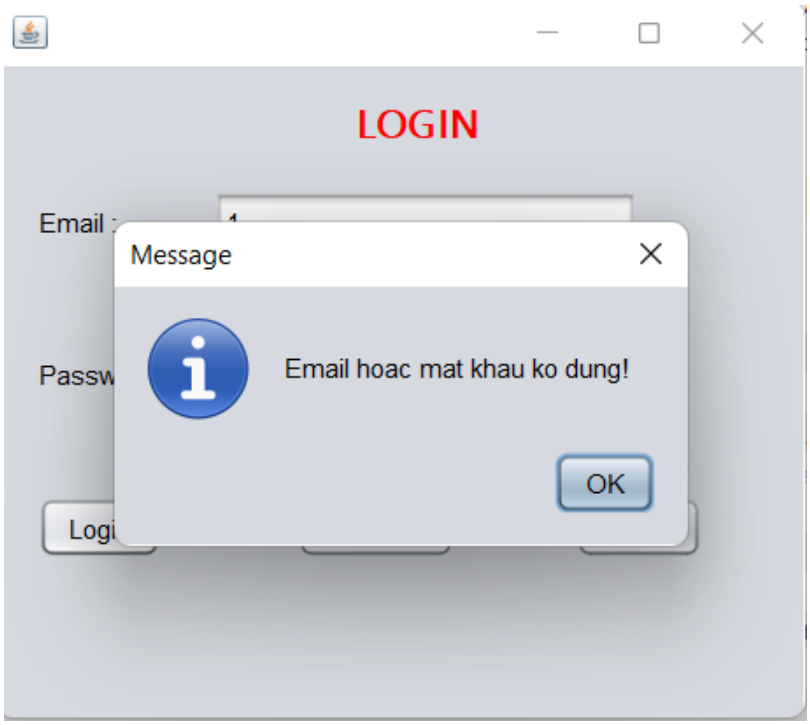
```
private void showCustomers(Customer customer) {
    var row = new Object[]{ //row: biến trung gian
        index++, customer.getId(), customer.getName(), customer.getKindofroom(), customer.getAge(),
                 customer.getGender(), customer.getDay(), customer.getMoney(), customer.getRoom()};
    tableModel.addRow(row);
    tableModel.fireTableDataChanged();

}
```

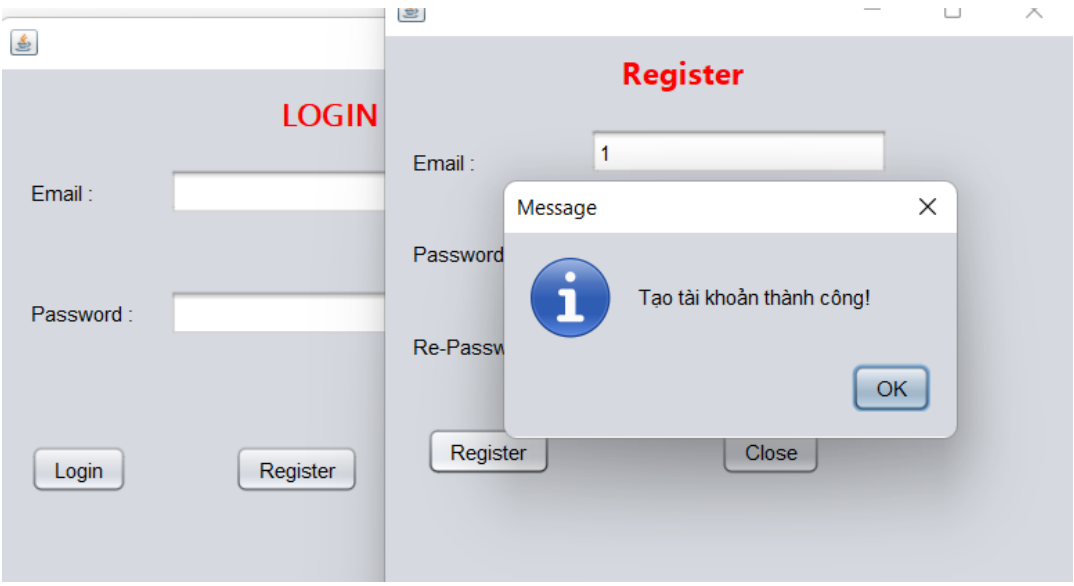
- Exit
  Exit system

```
private void btnCloseActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

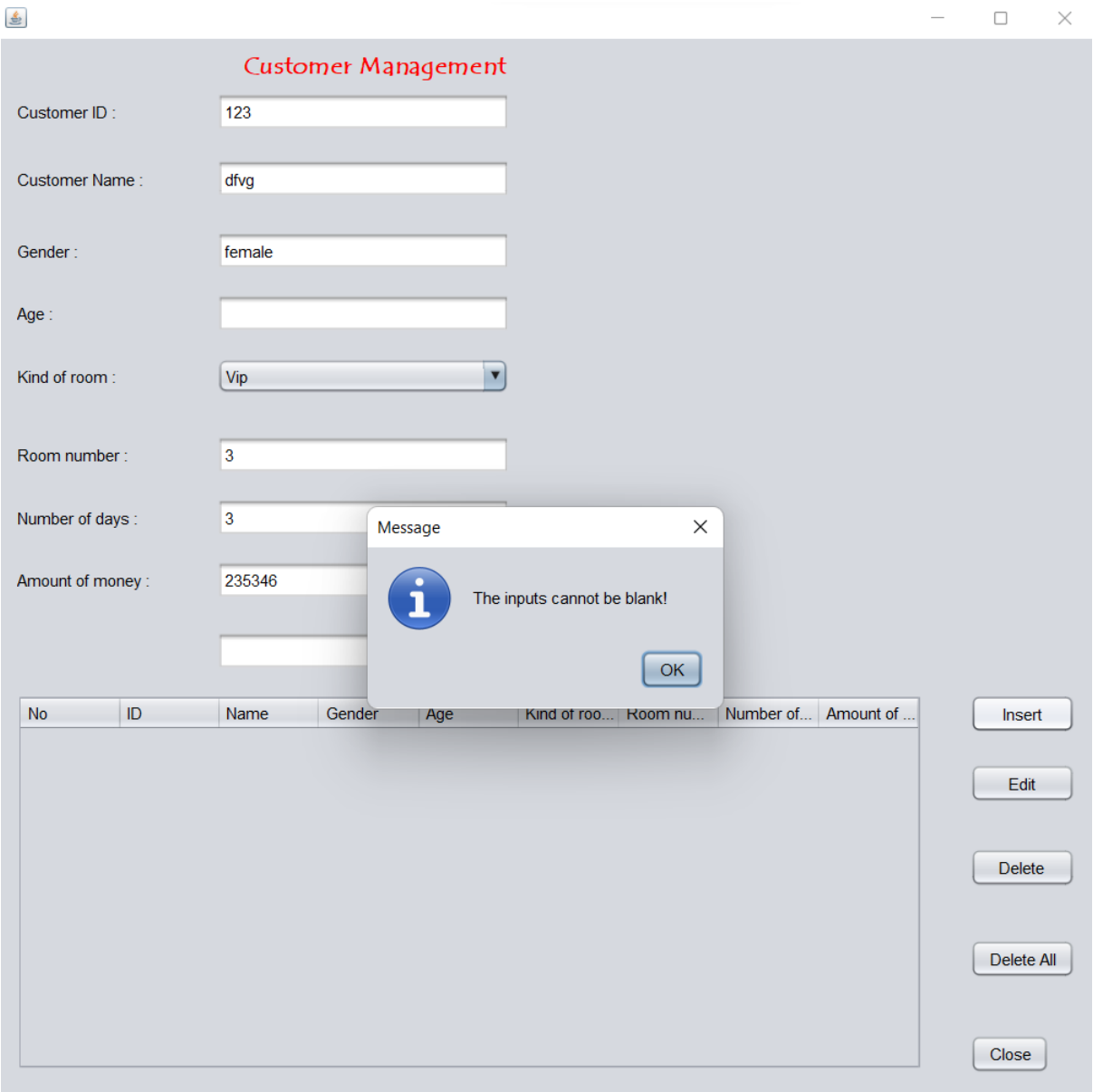
4. Explain how to handle errors
   - When a user logs into the account the following error may occur:



- To fix the error the user needs to register a new account.



- User forgot to enter age



V. Test

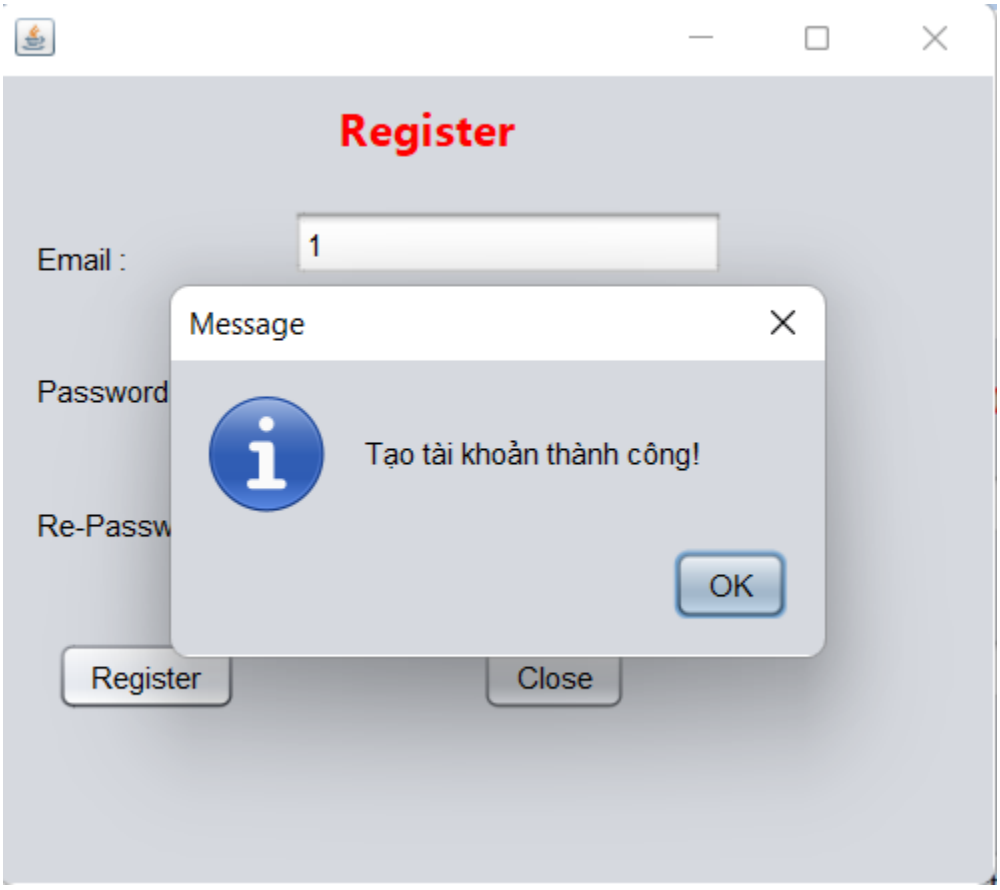
Test	Test Description	Test Steps	Test Data	Expected Results	Actural Results	Pass/False
1	Register	1/ Enter 2/ Login	- Email: 1 - Password: * - Re-Password: *	Successfully registered an account.	Successfull	Pass
2	Login	1/ Register 2/ Login	- Email: 1 - Password: *	Successfully logged into the	Successfull	Pass

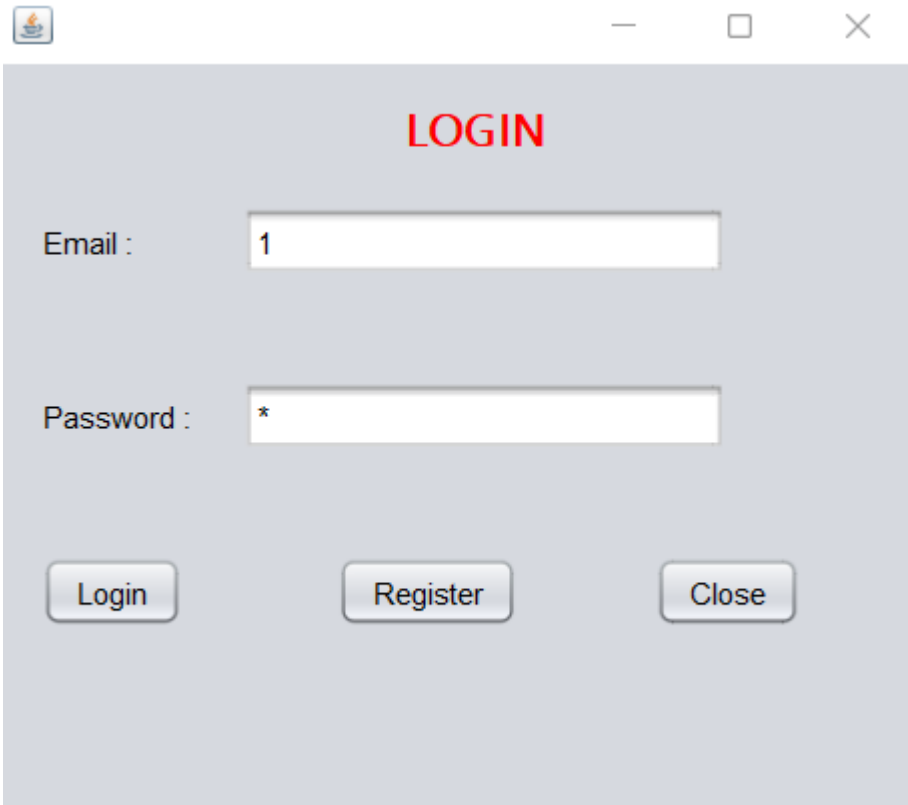
				account.		
3	Insert	1/ Enter 2/ Add customer	- Customer ID: 08 - Customer Name: Linh - Gender: Female - Age: 20 - Kind of room: President - Room number: 8 - Number of days: 2 - Amount of money: 520.000.000	Add customer	Successfull	Pass
4	Edit	1/ Enter 2/ Edit customer 3/ Update customer	- Customer ID: 28 - Customer Name: Anh - Gender: Male - Age: 21 - Kind of room: Vip - Room number: 3 - Number of days: 1 - Amount of money: 5.200.000	Edit id: 20 -> id: 28 and Update successfully.	Successfull	Pass
5	Insert new	1/ Enter 2/ Insert new customer	- Customer ID: 20 - Customer Name: Anh - Gender: Male - Age: 21 - Kind of room: Vip - Room number: 3 - Number of days: 1 - Amount of money: 5.200.000	Successfully inserted new more customers.	Successfull	Pass

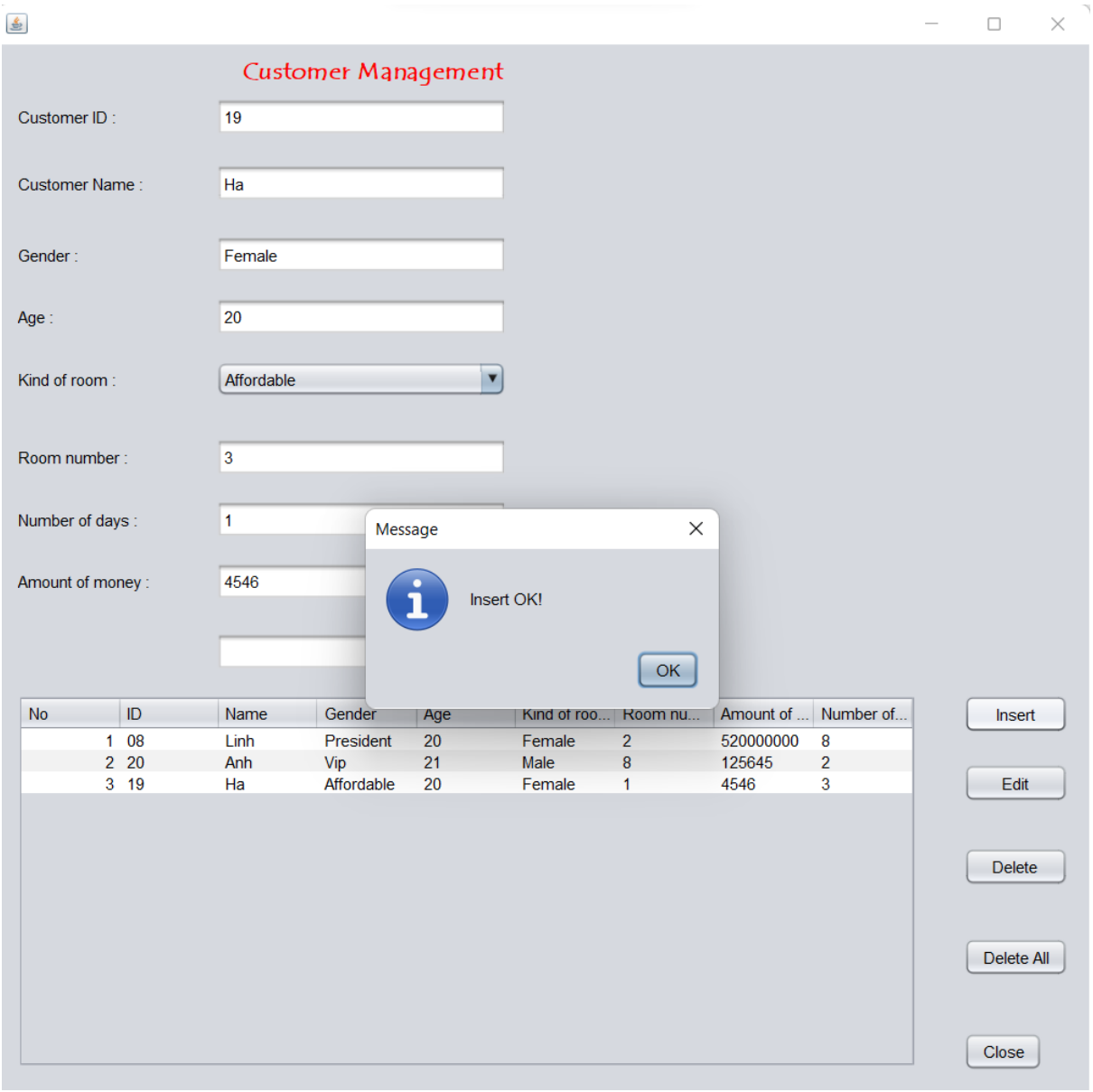
6	Delete	1/ Enter 2/ Delete customer	- Customer ID: 08 - Customer Name: Linh - Gender: Female - Age: 20 - Kind of room: President - Room number: 8 - Number of days: 2 - Amount of money: 520.000.000	Customers with attributes like: id, name, gender, age..... have been removed from the table.	Successfull	Pass
7	Delete All	1/ Enter 2/ Clear	- Customer ID: 08 - Customer Name: Linh - Gender: Female - Age: 20 - Kind of room: President - Room number: 8 - Number of days: 2 - Amount of money: 520.000.000 - Customer ID: 20 - Customer Name: Anh - Gender: Male - Age: 21 - Kind of room: Vip - Room number: 3 - Number of days: 1 - Amount of money: 5.200.000	All customers have been cleared.	Successfull	Pass

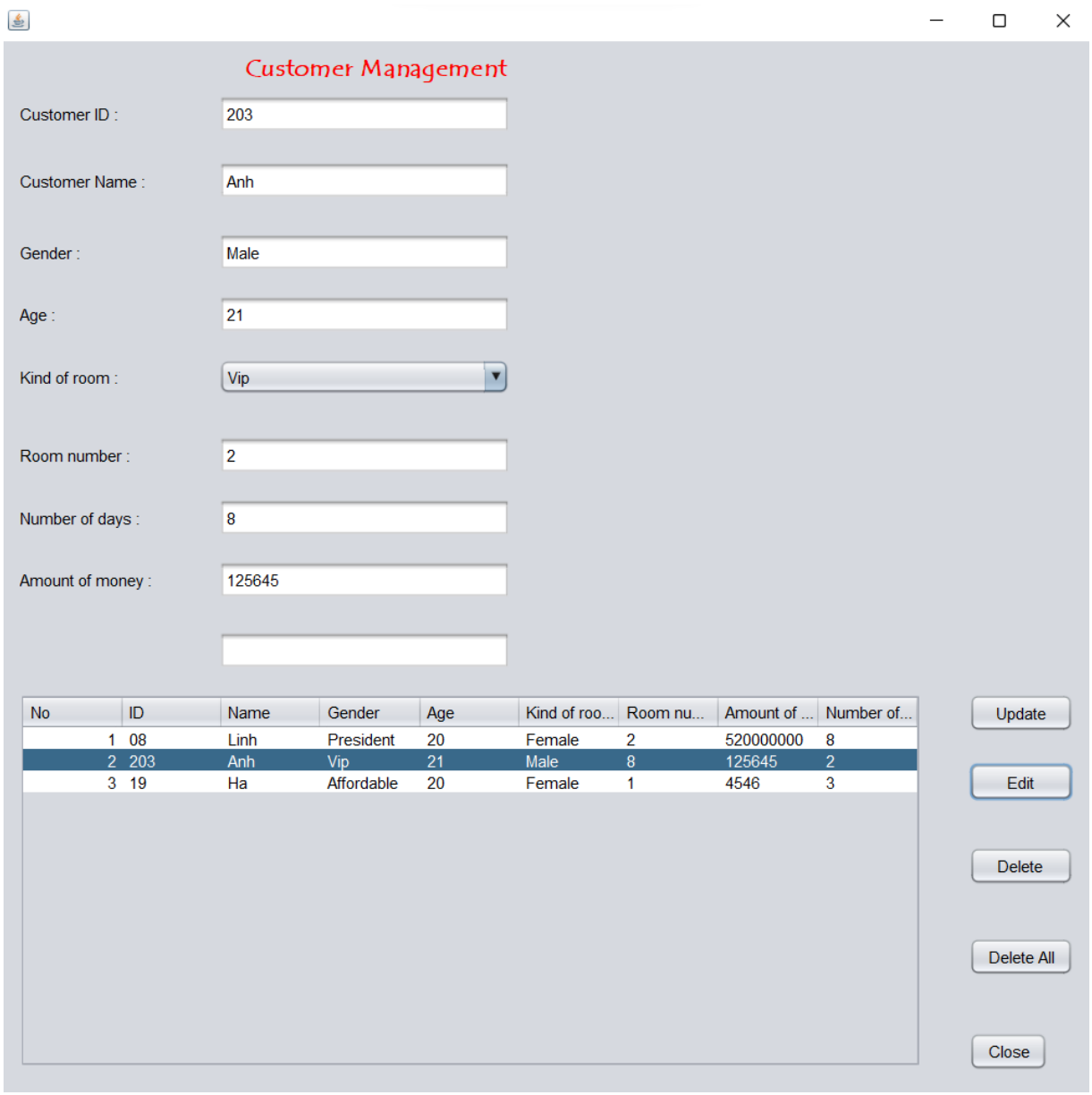
8	Search	1/ Enter ID 2/ Search	- Customer ID: 08 - Customer Name: Linh - Gender: Female - Age: 20 - Kind of room: President - Room number: 8 - Number of days: 2 - Amount of money: 520.000.000	Show ID, name, gender, age, kind of room, room number,… searched.	Successfull	Pass
9	Close	1/ Enter 2/ Exit	0	Exit program	Successfull	Pass

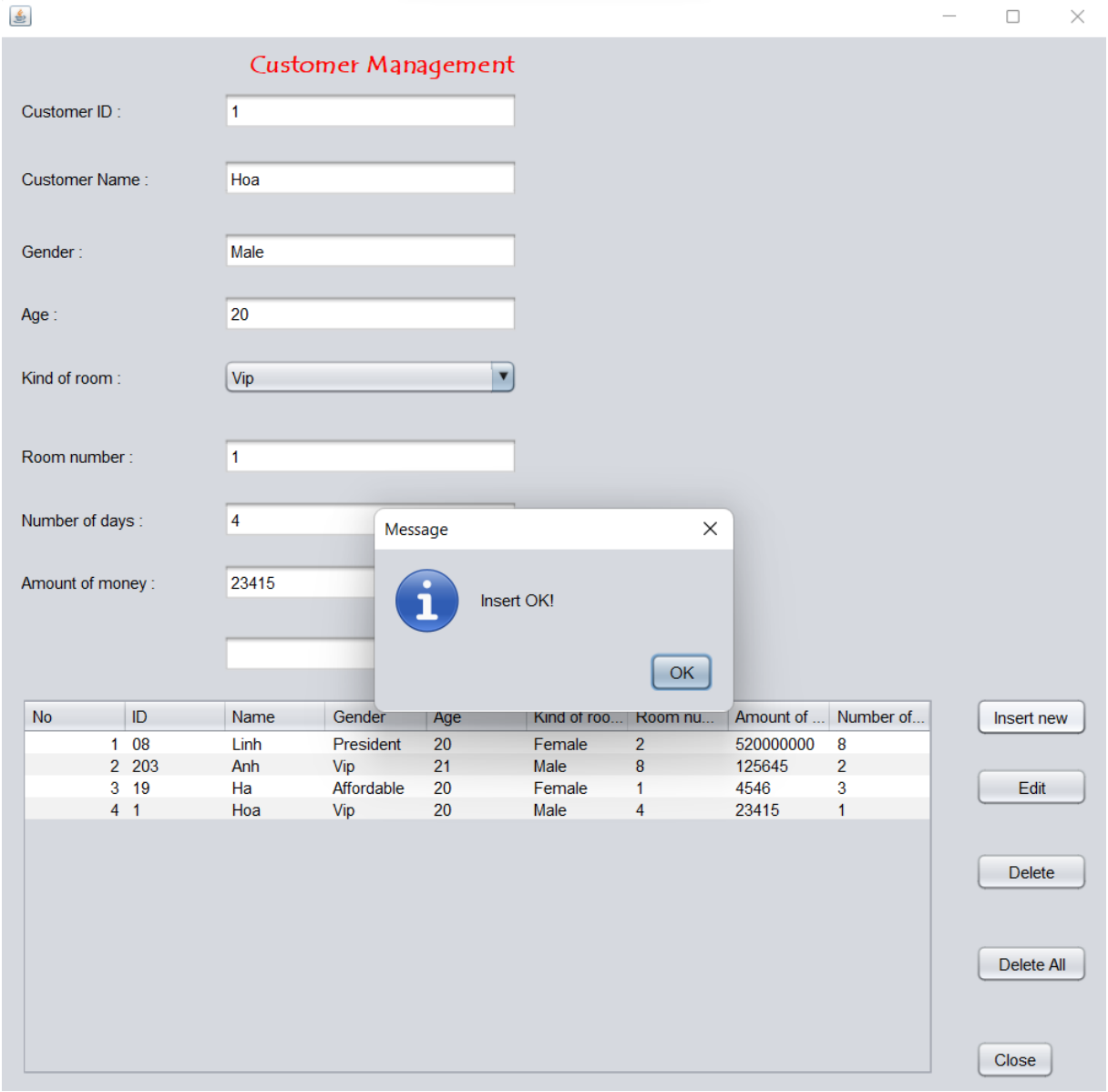
VI. Result

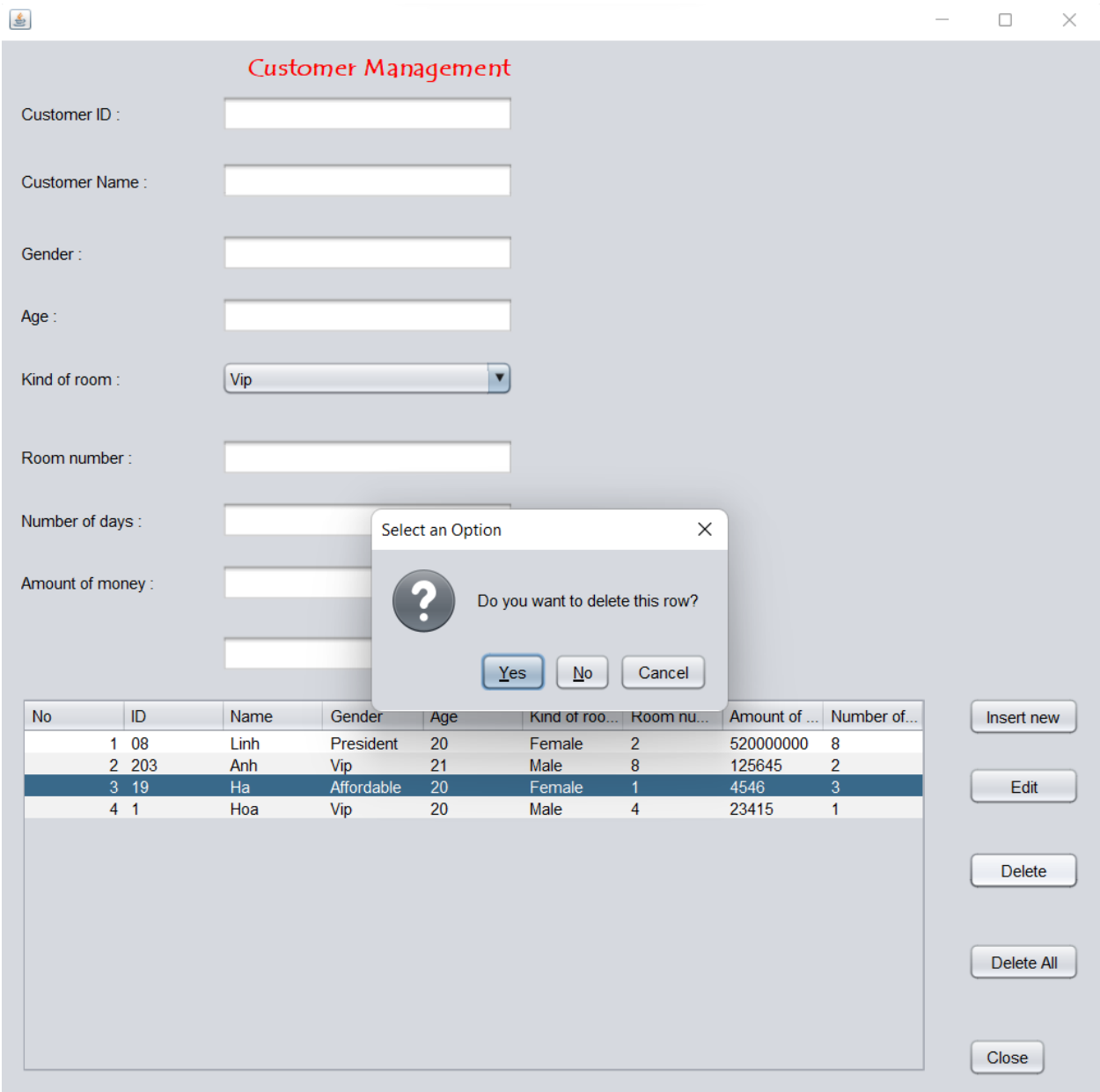

Register
Email :
1
Message
Tạo tài khoản thành công!
OK
Password
Re-Passw
Register
Close

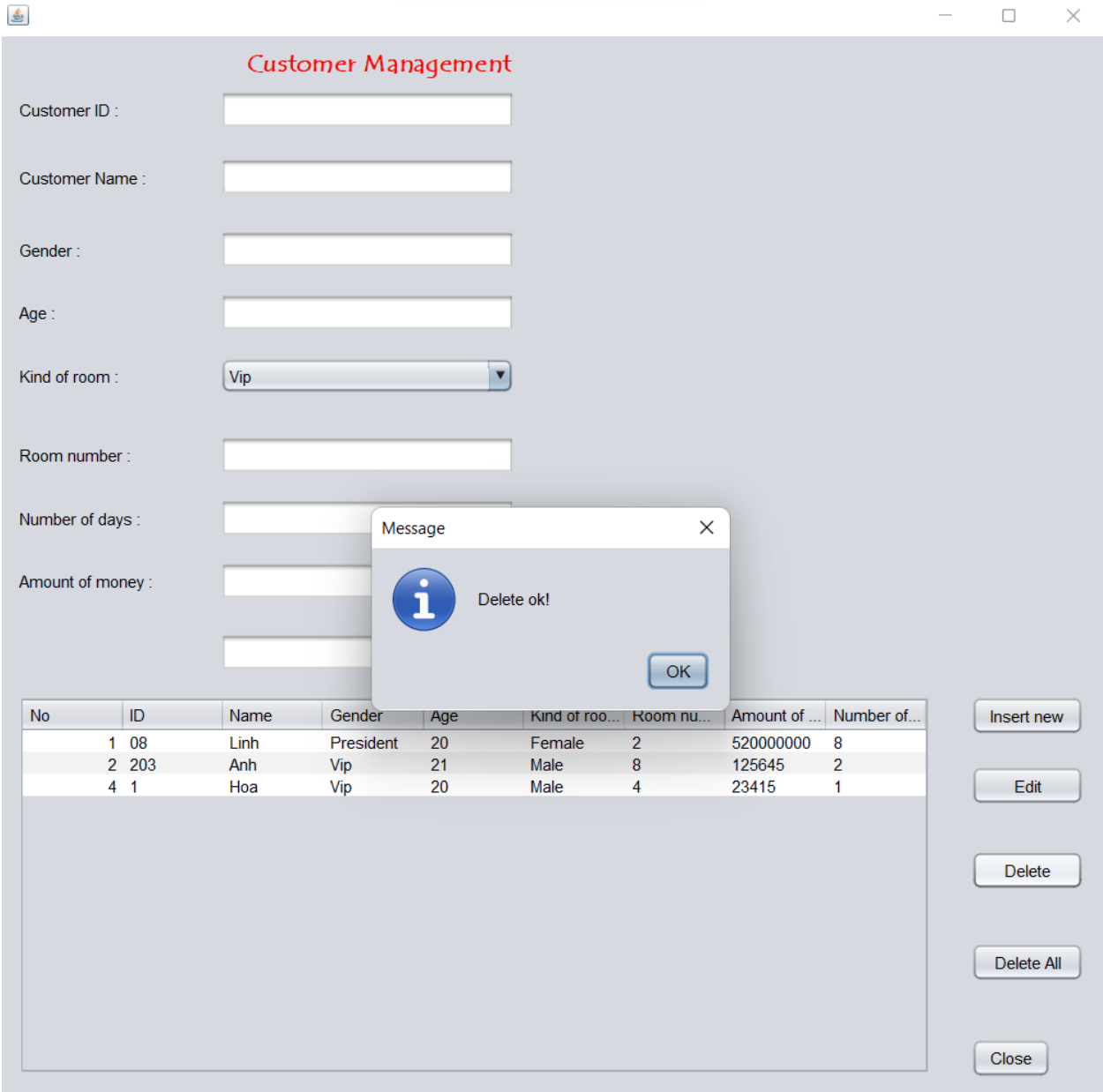

LOGIN
Email :
1
Password :
*
Login
Register
Close

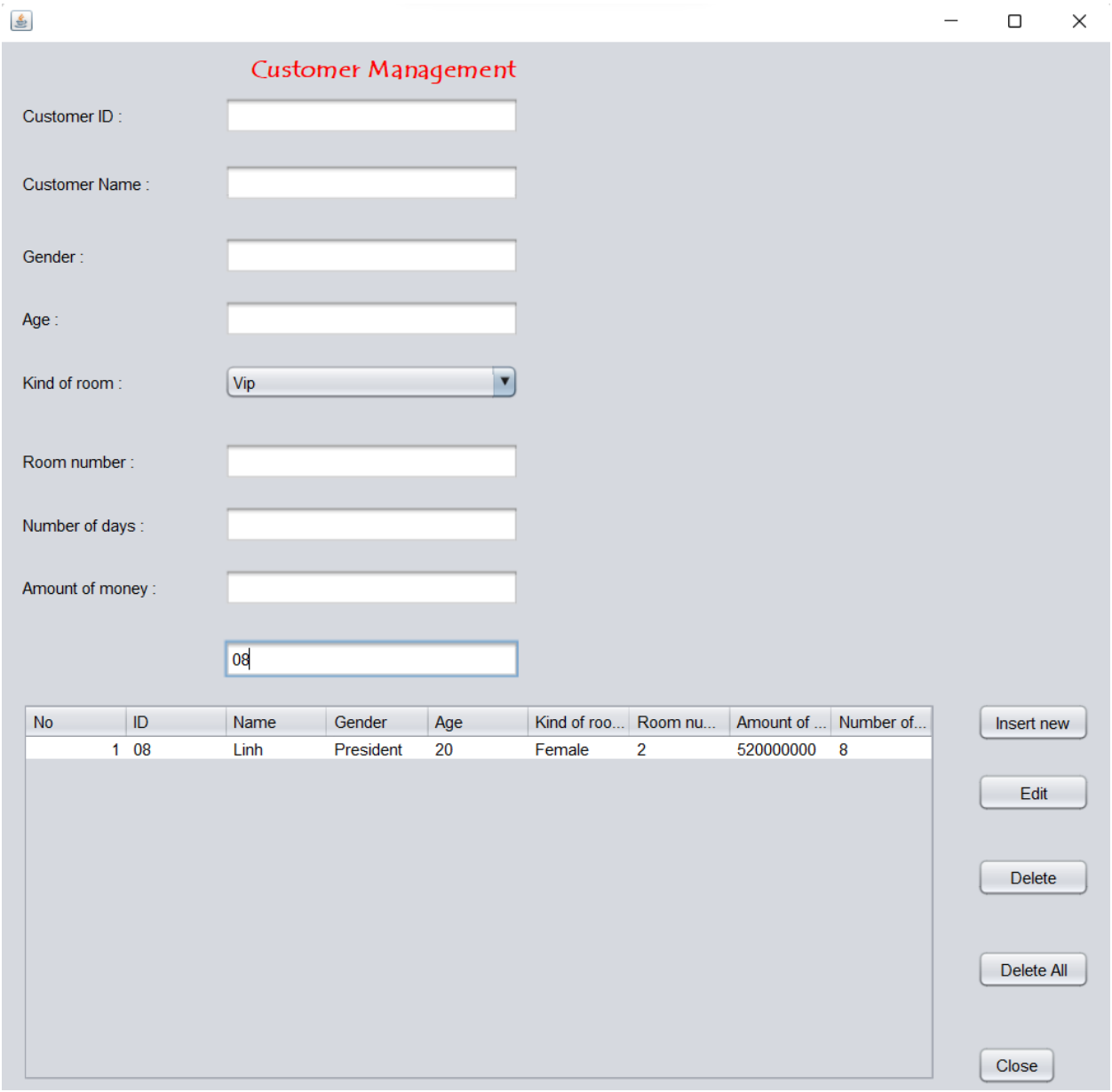

Customer Management
Customer ID :
19
Customer Name :
Ha
Gender :
Female
Age :
20
Kind of room :
Affordable
Room number :
3
Number of days :
1
Amount of money :
4546
Message
Insert OK!
OK
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1 08 Linh President 20 Female 2 520000000 8
2 20 Anh Vip 21 Male 8 125645 2
3 19 Ha Affordable 20 Female 1 4546 3
Insert
Edit
Delete
Delete All
Close

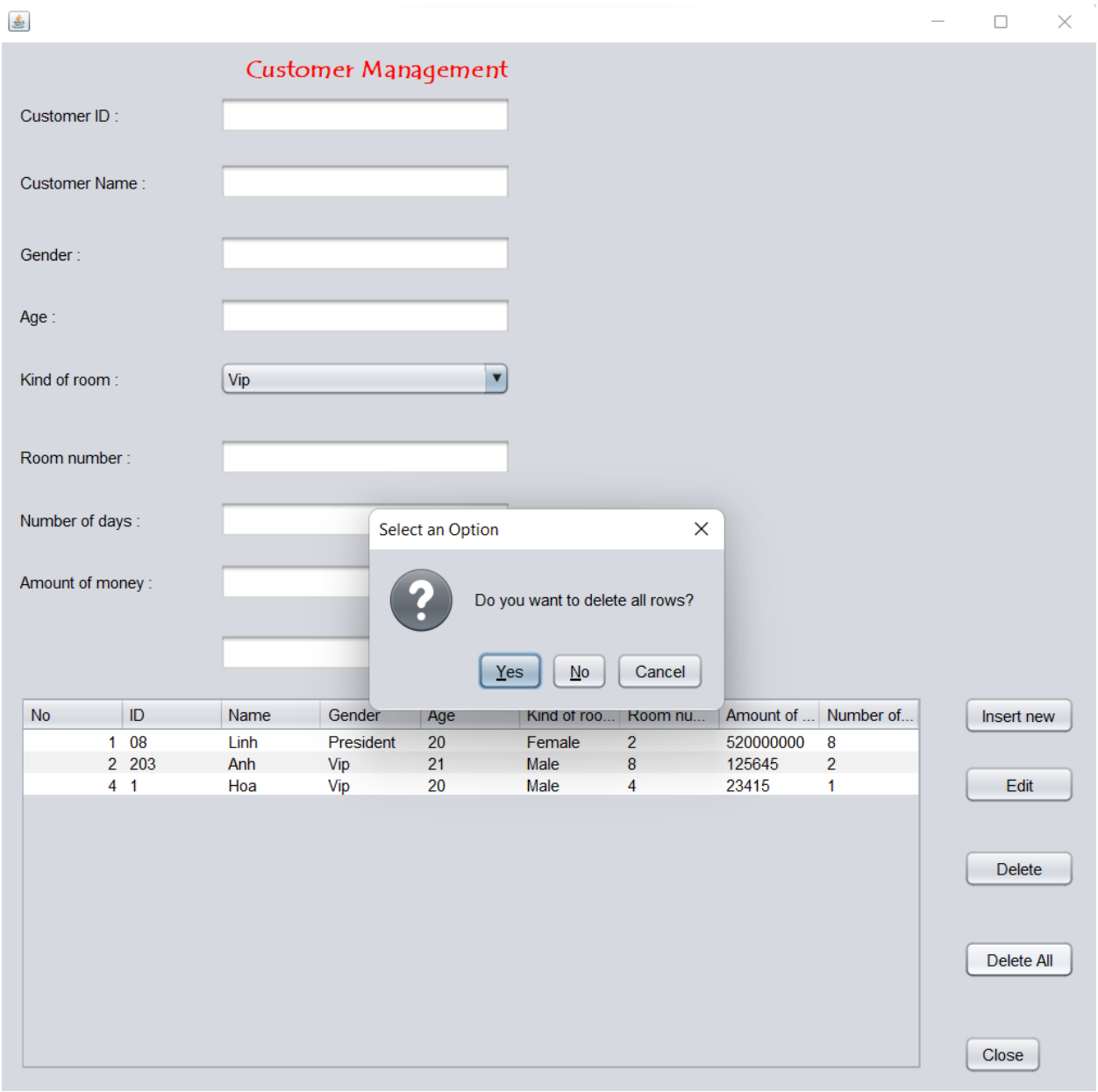

Customer Management
Customer ID :
203
Customer Name :
Anh
Gender :
Male
Age :
21
Kind of room :
Vip
Room number :
2
Number of days :
8
Amount of money :
125645
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1 08 Linh President 20 Female 2 520000000 8
2 203 Anh Vip 21 Male 8 125645 2
3 19 Ha Affordable 20 Female 1 4546 3
Update
Edit
Delete
Delete All
Close

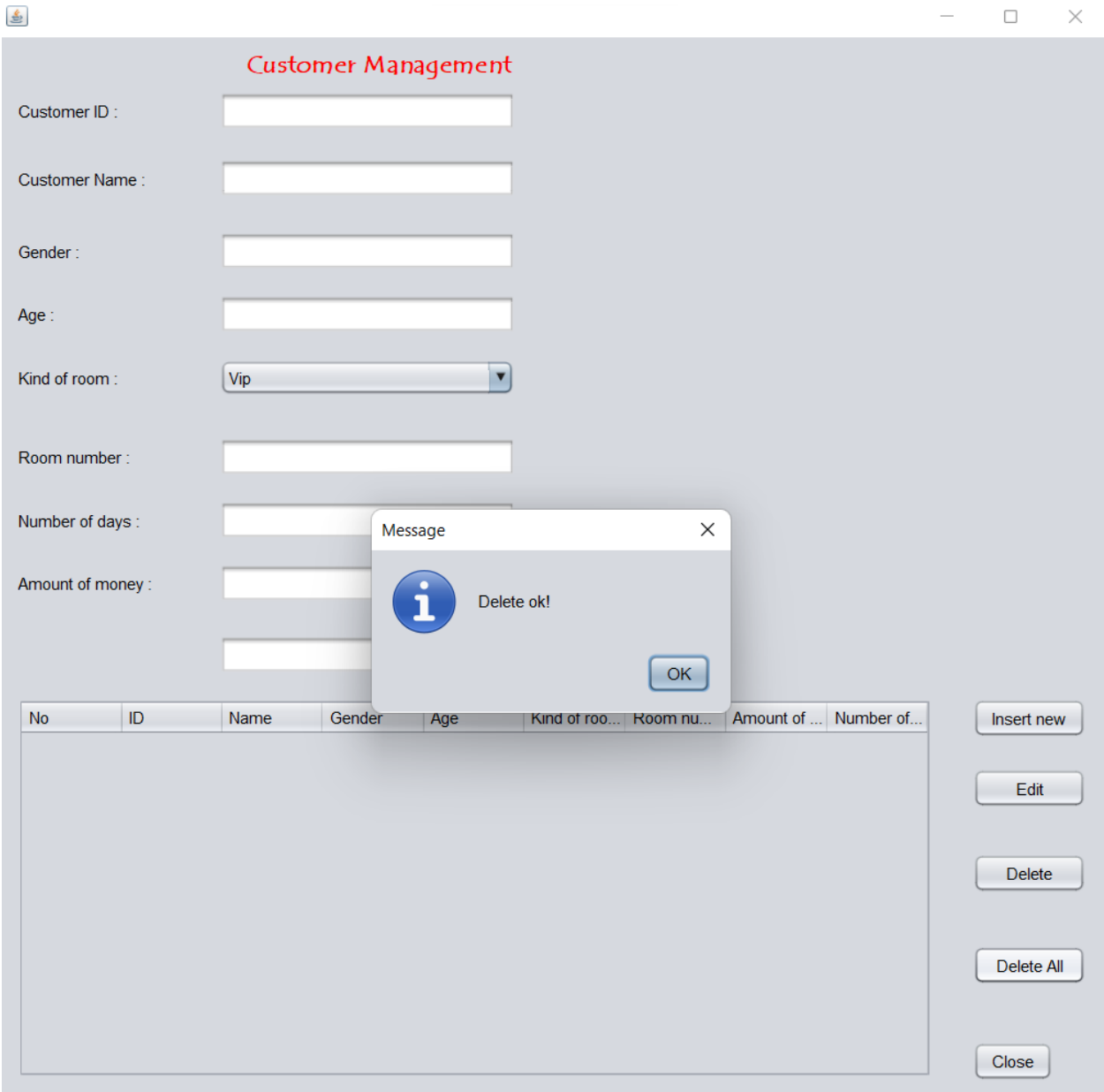

Customer Management
Customer ID :
1
Customer Name :
Hoa
Gender :
Male
Age :
20
Kind of room :
Vip
Room number :
1
Number of days :
4
Amount of money :
23415
Message
Insert OK!
OK
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1 08 Linh President 20 Female 2 520000000 8
2 203 Anh Vip 21 Male 8 125645 2
3 19 Ha Affordable 20 Female 1 4546 3
4 1 Hoa Vip 20 Male 4 23415 1
Insert new
Edit
Delete
Delete All
Close


Customer Management
Customer ID :
Customer Name :
Gender :
Age :
Kind of room :
Vip
Room number :
Number of days :
Amount of money :
Select an Option
Do you want to delete this row?
Yes
No
Cancel
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1 08 Linh President 20 Female 2 520000000 8
2 203 Anh Vip 21 Male 8 125645 2
3 19 Ha Affordable 20 Female 1 4546 3
4 1 Hoa Vip 20 Male 4 23415 1
Insert new
Edit
Delete
Delete All
Close


Customer Management
Customer ID :
Customer Name :
Gender :
Age :
Kind of room :
Vip
Room number :
Number of days :
Amount of money :
Message
Delete ok!
OK
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1 08 Linh President 20 Female 2 520000000 8
2 203 Anh Vip 21 Male 8 125645 2
4 1 Hoa Vip 20 Male 4 23415 1
Insert new
Edit
Delete
Delete All
Close


Customer Management
Customer ID :
Customer Name :
Gender :
Age :
Kind of room :
Vip
Room number :
Number of days :
Amount of money :
08
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1
08
Linh
President
20
Female
2
520000000
8
Insert new
Edit
Delete
Delete All
Close


Customer Management
Customer ID :
Customer Name :
Gender :
Age :
Kind of room :
Vip
Room number :
Number of days :
Amount of money :
Select an Option
Do you want to delete all rows?
Yes
No
Cancel
No
ID
Name
Gender
Age
Kind of roo...
Room nu...
Amount of ...
Number of...
1 08 Linh President 20 Female 2 520000000 8
2 203 Anh Vip 21 Male 8 125645 2
4 1 Hoa Vip 20 Male 4 23415 1
Insert new
Edit
Delete
Delete All
Close



VII. Conclusion

My program is complete and can help users manage customer information. This program has the functions of adding, updating, editing and deleting information. There is also a search function to help users control information more easily. However, there are still some limitations that need to be overcome such as: the interface is not attractive to viewers, lack of many functions, ... If given the opportunity, I will make the program to manage more elements and better interface.