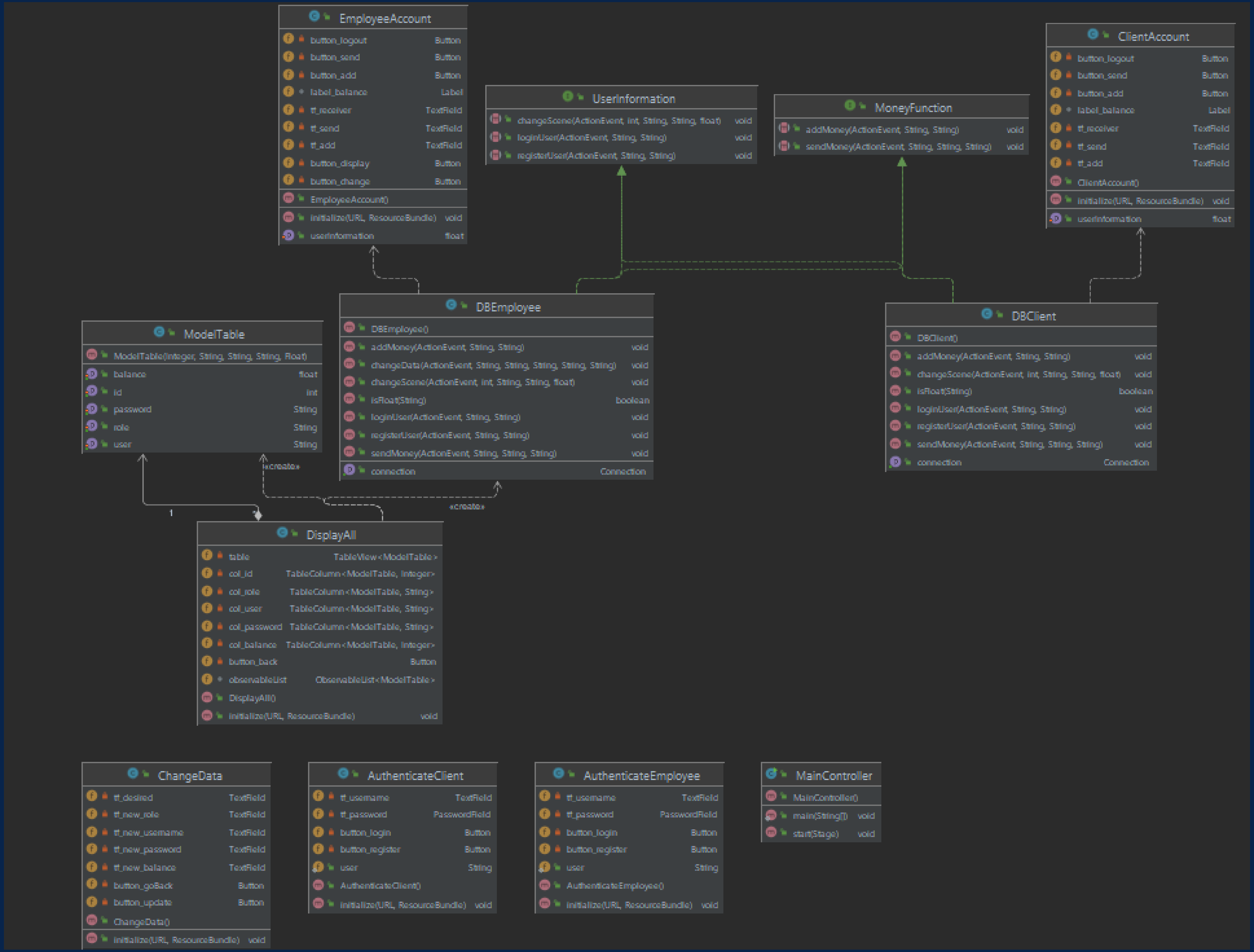
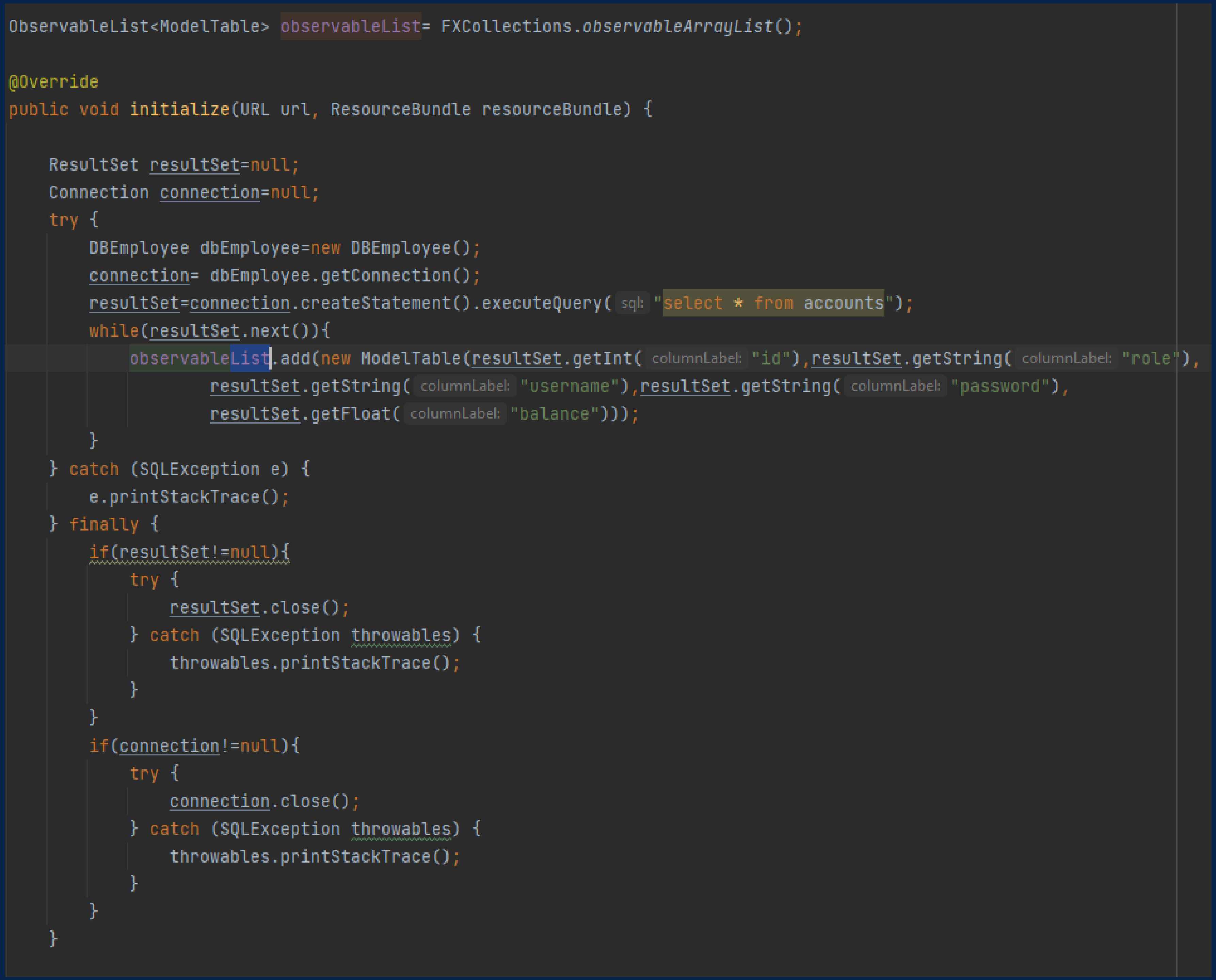
Bank Account

Andrei Bogdanescu
West University of Timisoara

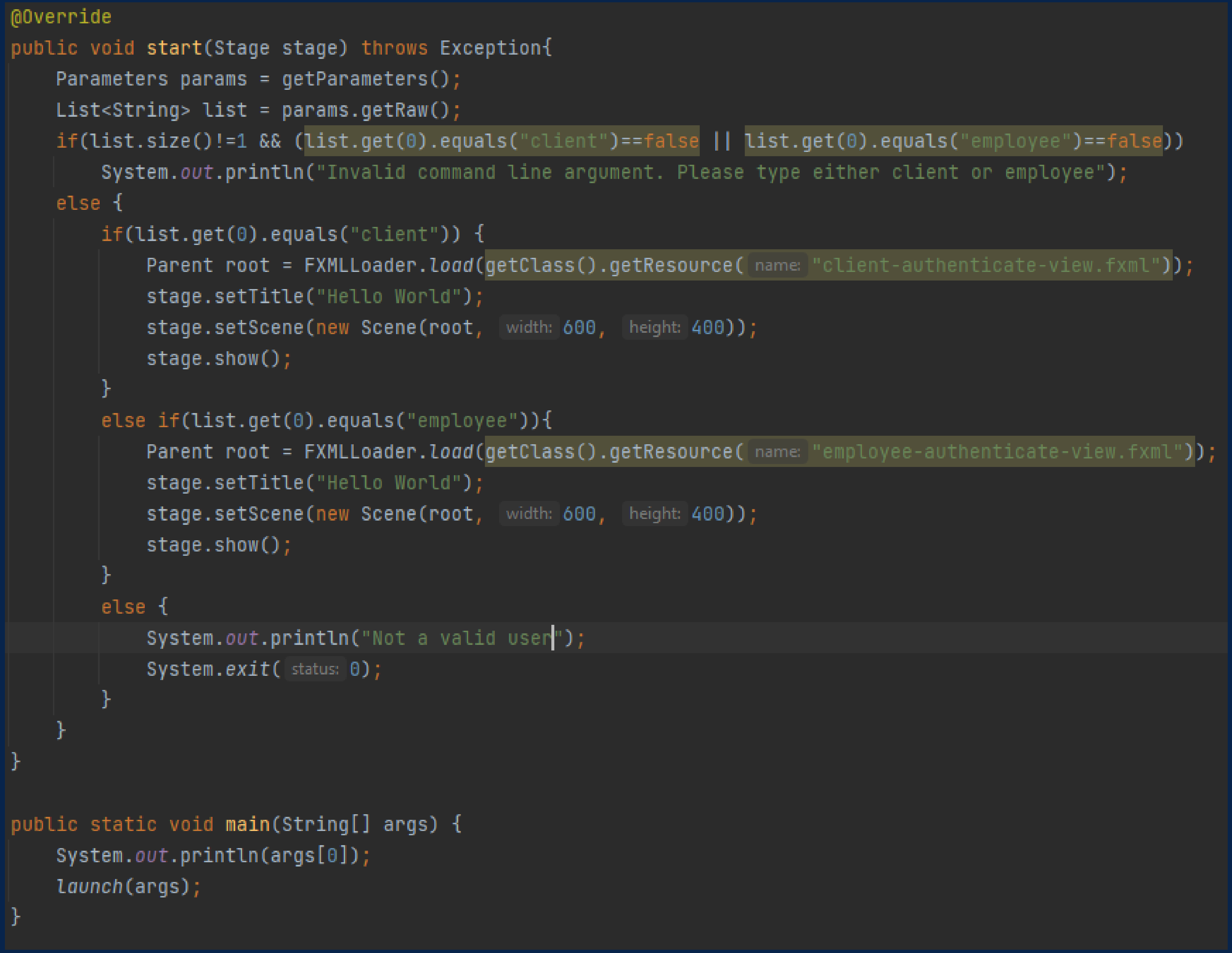
Why did I choose this project?

Model Requirement 1


EmployeeAccount
button_logout Button
button_send Button
button_add Button
label_balance Label
tf_receiver TextField
tf_send TextField
tf_add TextField
button_display Button
button_change Button
EmployeeAccount()
initialize(URL, ResourceBundle) void
userInformation float
UserInformation
changeScene(ActionEvent, int, String, String, float) void
loginUser(ActionEvent, String, String) void
registerUser(ActionEvent, String, String) void
MoneyFunction
addMoney(ActionEvent, String, String) void
sendMoney(ActionEvent, String, String, String) void
ClientAccount
button_logout Button
button_send Button
button_add Button
label_balance Label
tf_receiver TextField
tf_send TextField
tf_add TextField
ClientAccount()
initialize(URL, ResourceBundle) void
userInformation float
ModelTable
ModelTable(Integer, String, String, String, Float)
balance float
id int
password String
role String
user String
DBEmployee
DBEmployee()
addMoney(ActionEvent, String, String) void
changeData(ActionEvent, String, String, String, String, String) void
changeScene(ActionEvent, int, String, String, float) void
isFloat(String) boolean
loginUser(ActionEvent, String, String) void
registerUser(ActionEvent, String, String) void
sendMoney(ActionEvent, String, String, String) void
connection Connection
DBClient
DBClient()
addMoney(ActionEvent, String, String) void
changeScene(ActionEvent, int, String, String, float) void
isFloat(String) boolean
loginUser(ActionEvent, String, String) void
registerUser(ActionEvent, String, String) void
sendMoney(ActionEvent, String, String, String) void
connection Connection
«create»
1
«create»
DisplayAll
table TableView<ModelTable>
col_id TableColumn<ModelTable, Integer>
col_role TableColumn<ModelTable, String>
col_user TableColumn<ModelTable, String>
col_password TableColumn<ModelTable, String>
col_balance TableColumn<ModelTable, Integer>
button_back Button
observableList ObservableList<ModelTable>
DisplayAll()
initialize(URL, ResourceBundle) void
ChangeData
tf_desired TextField
tf_new_role TextField
tf_new_username TextField
tf_new_password TextField
tf_new_balance TextField
button_goBack Button
button_update Button
ChangeData()
initialize(URL, ResourceBundle) void
AuthenticateClient
tf_username TextField
tf_password PasswordField
button_login Button
button_register Button
user String
AuthenticateClient()
initialize(URL, ResourceBundle) void
AuthenticateEmployee
tf_username TextField
tf_password PasswordField
button_login Button
button_register Button
user String
AuthenticateEmployee()
initialize(URL, ResourceBundle) void
MainController
MainController()
main(String[]) void
start(Stage) void

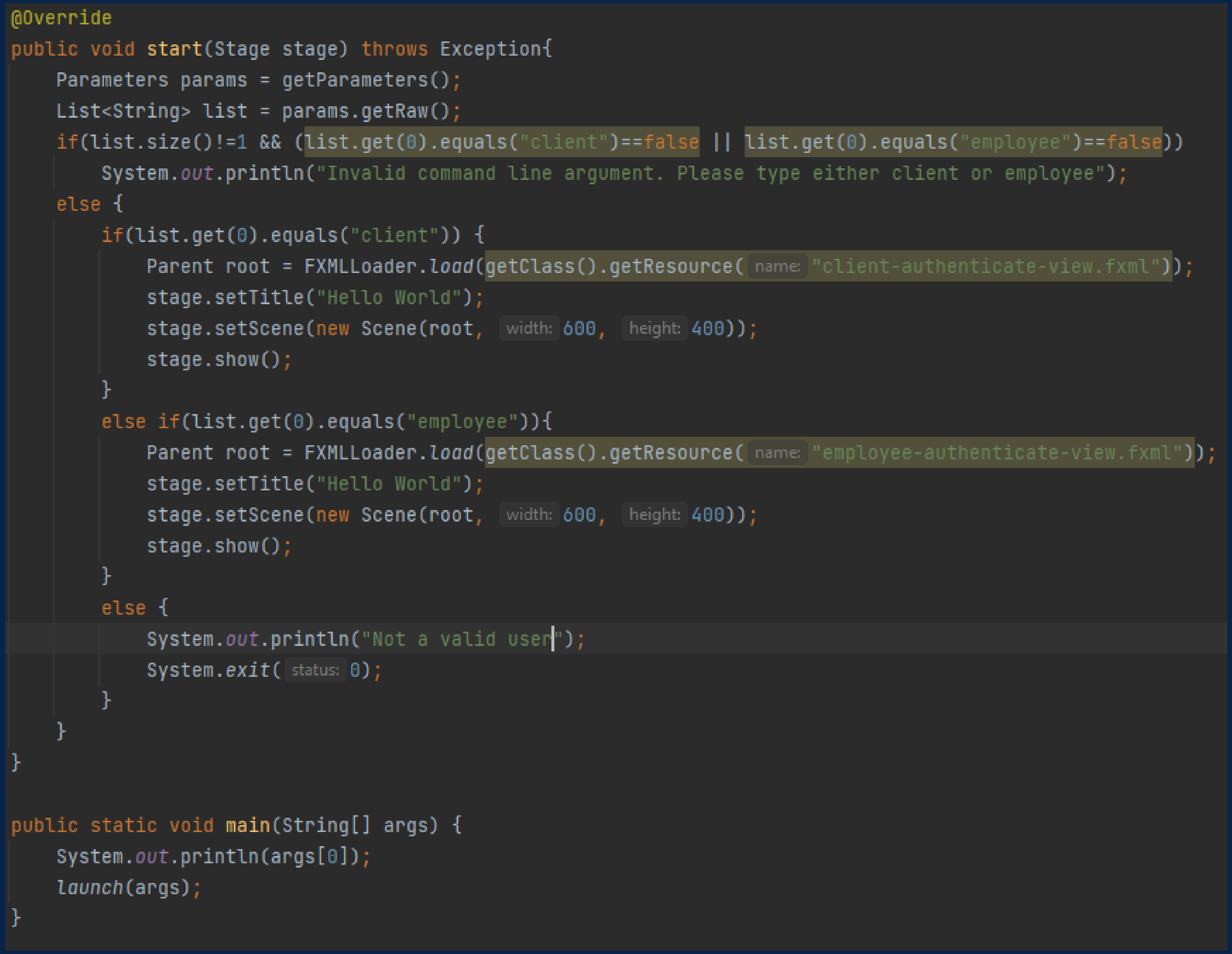


```java
ObservableList<ModelTable> observableList= FXCollections.observableArrayList();

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {

    ResultSet resultSet=null;
    Connection connection=null;
    try {
        DBEmployee dbEmployee=new DBEmployee();
        connection= dbEmployee.getConnection();
        resultSet=connection.createStatement().executeQuery( sql: "select * from accounts");
        while(resultSet.next()){
            observableList.add(new ModelTable(resultSet.getInt( columnLabel: "id"),resultSet.getString( columnLabel: "role"),
                    resultSet.getString( columnLabel: "username"),resultSet.getString( columnLabel: "password"),
                    resultSet.getFloat( columnLabel: "balance")));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if(resultSet!=null){
            try {
                resultSet.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
        if(connection!=null){
            try {
                connection.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
    }
```



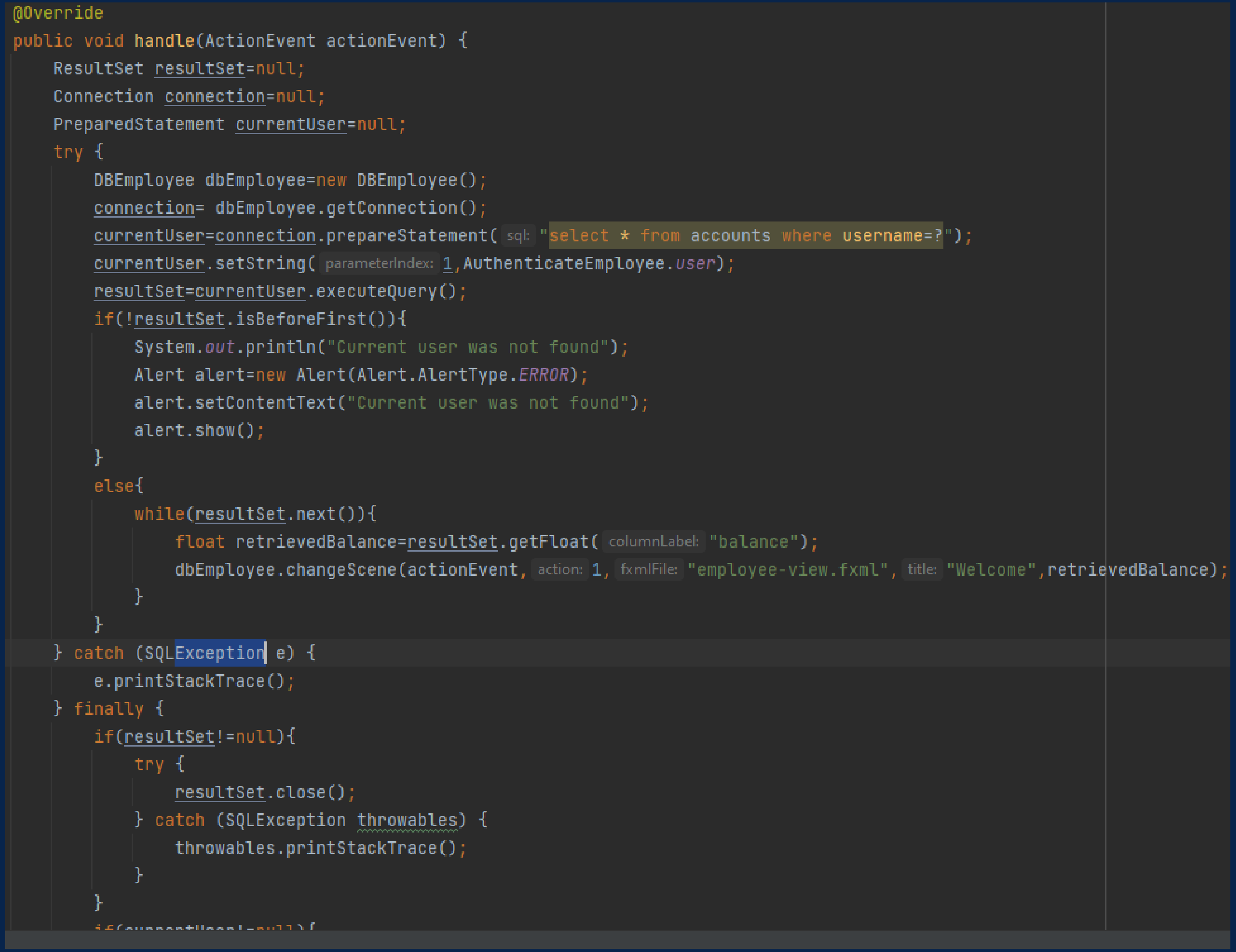
```java
@Override
public void start(Stage stage) throws Exception{
    Parameters params = getParameters();
    List<String> list = params.getRaw();
    if(list.size()!=1 && (list.get(0).equals("client")==false || list.get(0).equals("employee")==false))
        System.out.println("Invalid command line argument. Please type either client or employee");
    else {
        if(list.get(0).equals("client")) {
            Parent root = FXMLLoader.load(getClass().getResource( name: "client-authenticate-view.fxml"));
            stage.setTitle("Hello World");
            stage.setScene(new Scene(root,  width: 600,  height: 400));
            stage.show();
        }
        else if(list.get(0).equals("employee")){
            Parent root = FXMLLoader.load(getClass().getResource( name: "employee-authenticate-view.fxml"));
            stage.setTitle("Hello World");
            stage.setScene(new Scene(root,  width: 600,  height: 400));
            stage.show();
        }
        else {
            System.out.println("Not a valid user");
            System.exit( status: 0);
        }
    }
}

public static void main(String[] args) {
    System.out.println(args[0]);
    launch(args);
}
```

Requirement 2

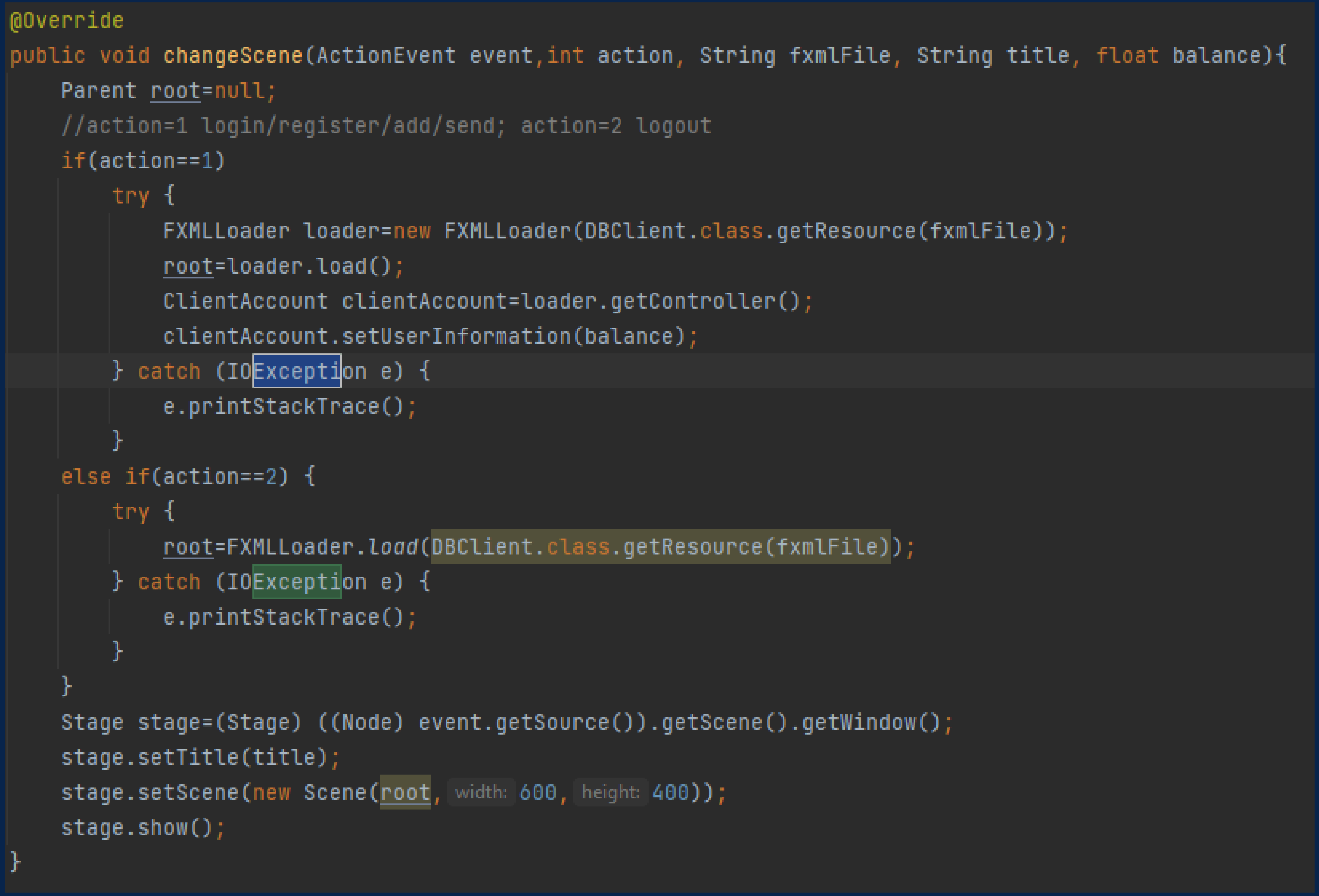


```java
@Override
public void start(Stage stage) throws Exception{
    Parameters params = getParameters();
    List<String> list = params.getRaw();
    if(list.size()!=1 && (list.get(0).equals("client")==false || list.get(0).equals("employee")==false))
        System.out.println("Invalid command line argument. Please type either client or employee");
    else {
        if(list.get(0).equals("client")) {
            Parent root = FXMLLoader.load(getClass().getResource( name: "client-authenticate-view.fxml"));
            stage.setTitle("Hello World");
            stage.setScene(new Scene(root,  width: 600,  height: 400));
            stage.show();
        }
        else if(list.get(0).equals("employee")){
            Parent root = FXMLLoader.load(getClass().getResource( name: "employee-authenticate-view.fxml"));
            stage.setTitle("Hello World");
            stage.setScene(new Scene(root,  width: 600,  height: 400));
            stage.show();
        }
        else {
            System.out.println("Not a valid user");
            System.exit( status: 0);
        }
    }
}

public static void main(String[] args) {
    System.out.println(args[0]);
    launch(args);
}
```

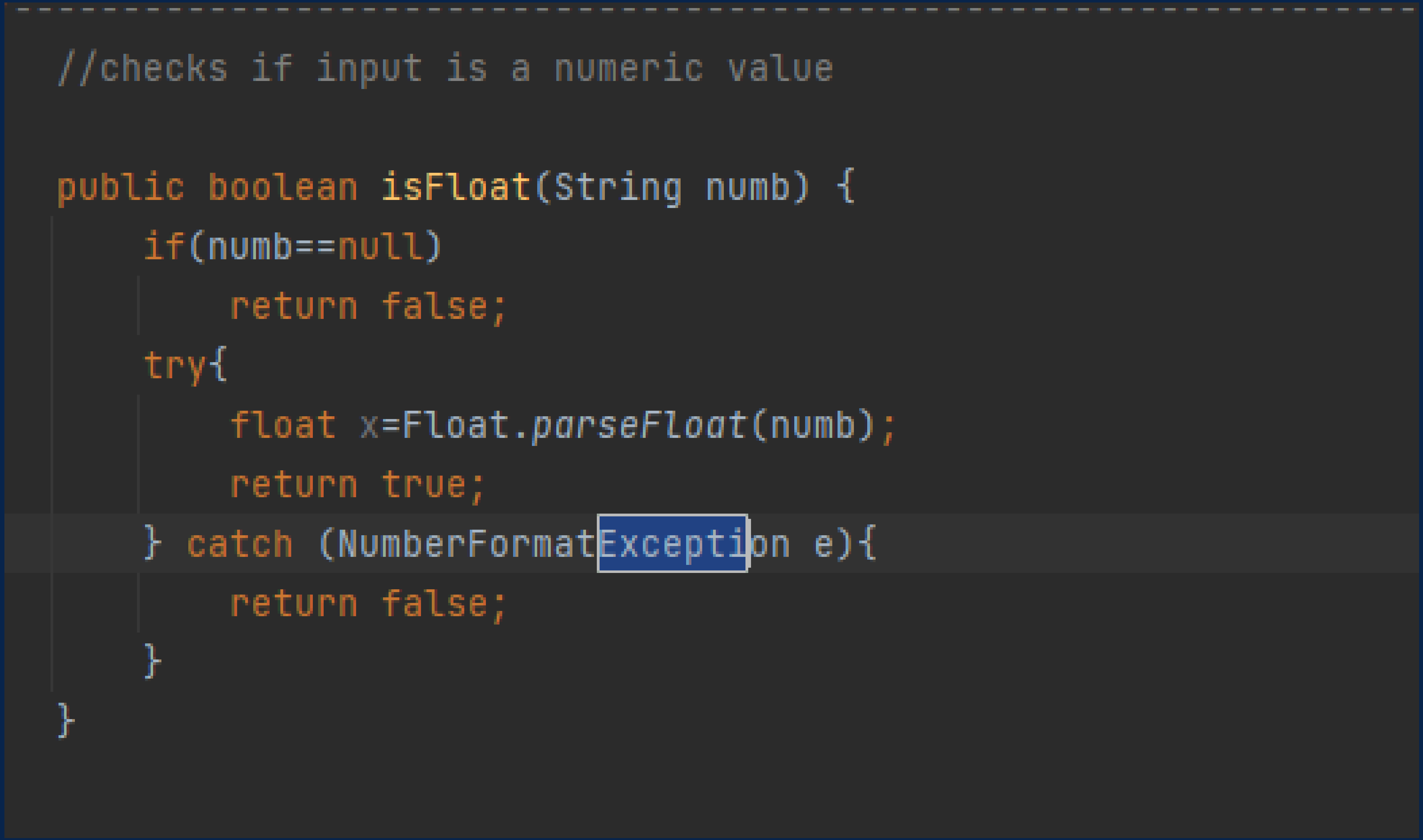
Requirement 4



```java
@Override
public void handle(ActionEvent actionEvent) {
    ResultSet resultSet=null;
    Connection connection=null;
    PreparedStatement currentUser=null;
    try {
        DBEmployee dbEmployee=new DBEmployee();
        connection= dbEmployee.getConnection();
        currentUser=connection.prepareStatement( sql: "select * from accounts where username=?");
        currentUser.setString( parameterIndex: 1,AuthenticateEmployee.user);
        resultSet=currentUser.executeQuery();
        if(!resultSet.isBeforeFirst()){
            System.out.println("Current user was not found");
            Alert alert=new Alert(Alert.AlertType.ERROR);
            alert.setContentText("Current user was not found");
            alert.show();
        }
        else{
            while(resultSet.next()){
                float retrievedBalance=resultSet.getFloat( columnLabel: "balance");
                dbEmployee.changeScene(actionEvent, action: 1, fxmlFile: "employee-view.fxml", title: "Welcome",retrievedBalance);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if(resultSet!=null){
            try {
                resultSet.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        }
```

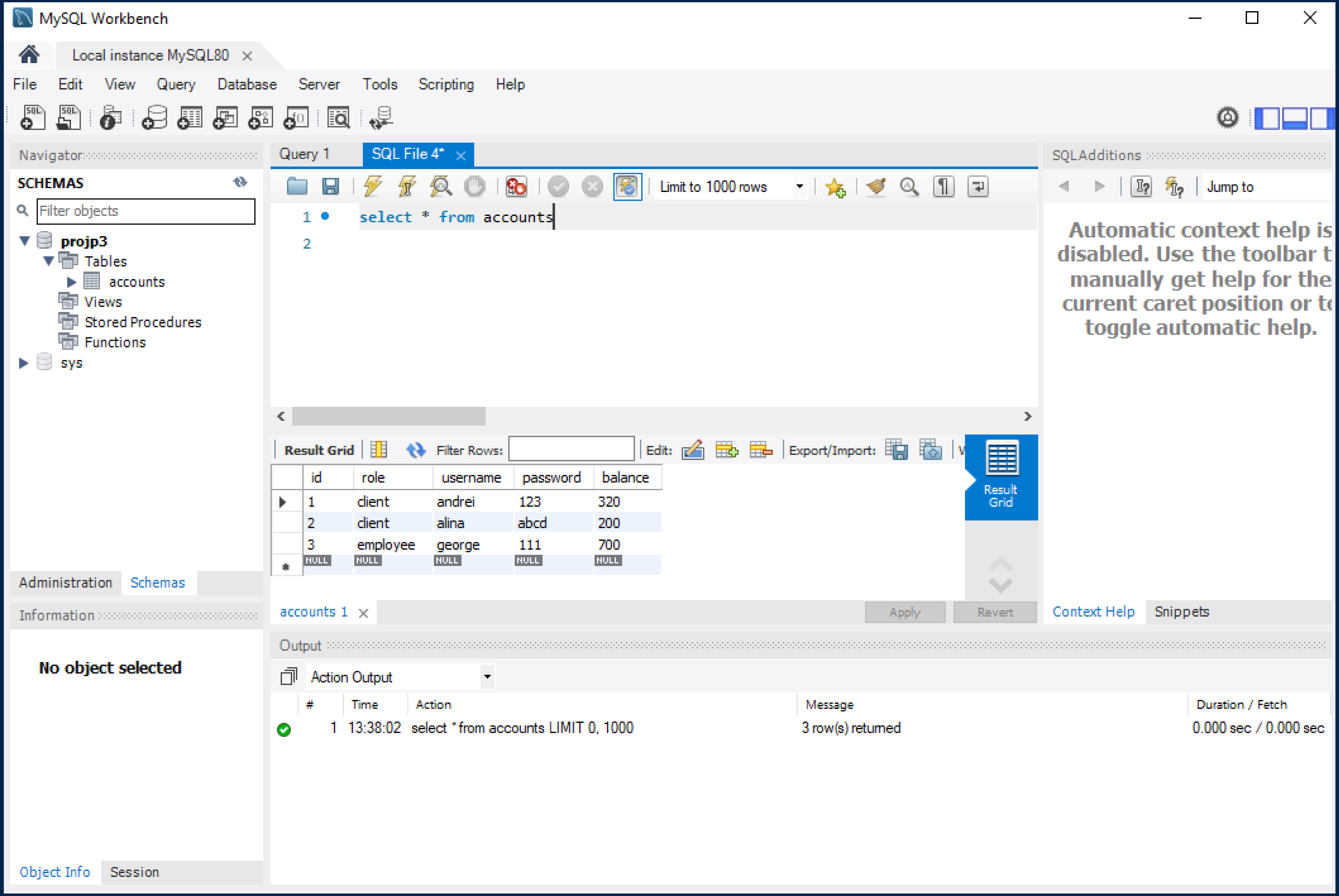


```java
@Override
public void changeScene(ActionEvent event,int action, String fxmlFile, String title, float balance){
    Parent root=null;
    //action=1 login/register/add/send; action=2 logout
    if(action==1)
        try {
            FXMLLoader loader=new FXMLLoader(DBClient.class.getResource(fxmlFile));
            root=loader.load();
            ClientAccount clientAccount=loader.getController();
            clientAccount.setUserInformation(balance);
        } catch (IOException e) {
            e.printStackTrace();
        }
    else if(action==2) {
        try {
            root=FXMLLoader.load(DBClient.class.getResource(fxmlFile));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    Stage stage=(Stage) ((Node) event.getSource()).getScene().getWindow();
    stage.setTitle(title);
    stage.setScene(new Scene(root, width: 600, height: 400));
    stage.show();
}
```



```
//checks if input is a numeric value

public boolean isFloat(String numb) {
    if(numb==null)
        return false;
    try{
        float x=Float.parseFloat(numb);
        return true;
    } catch (NumberFormatException e){
        return false;
    }
}
```

Database Requirement 1



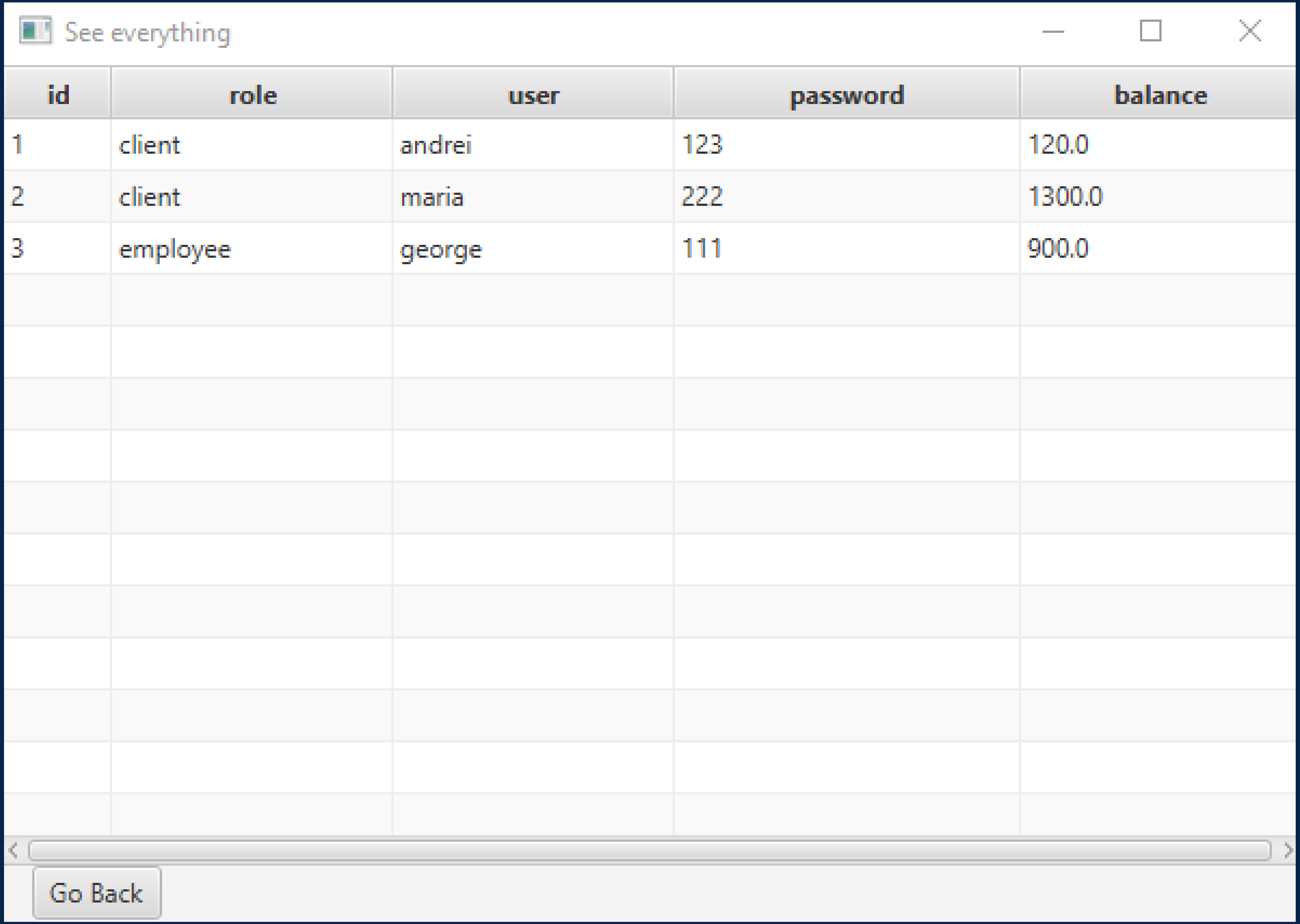
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

```
select * from accounts
```

id	role	username	password	balance
1	client	andrei	123	320
2	client	alina	abcd	200
3	employee	george	111	700
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                               Message             Duration / Fetch
1   13:38:02  select * from accounts LIMIT 0, 1000  3 row(s) returned   0.000 sec / 0.000 sec
```

Database Requirement 2

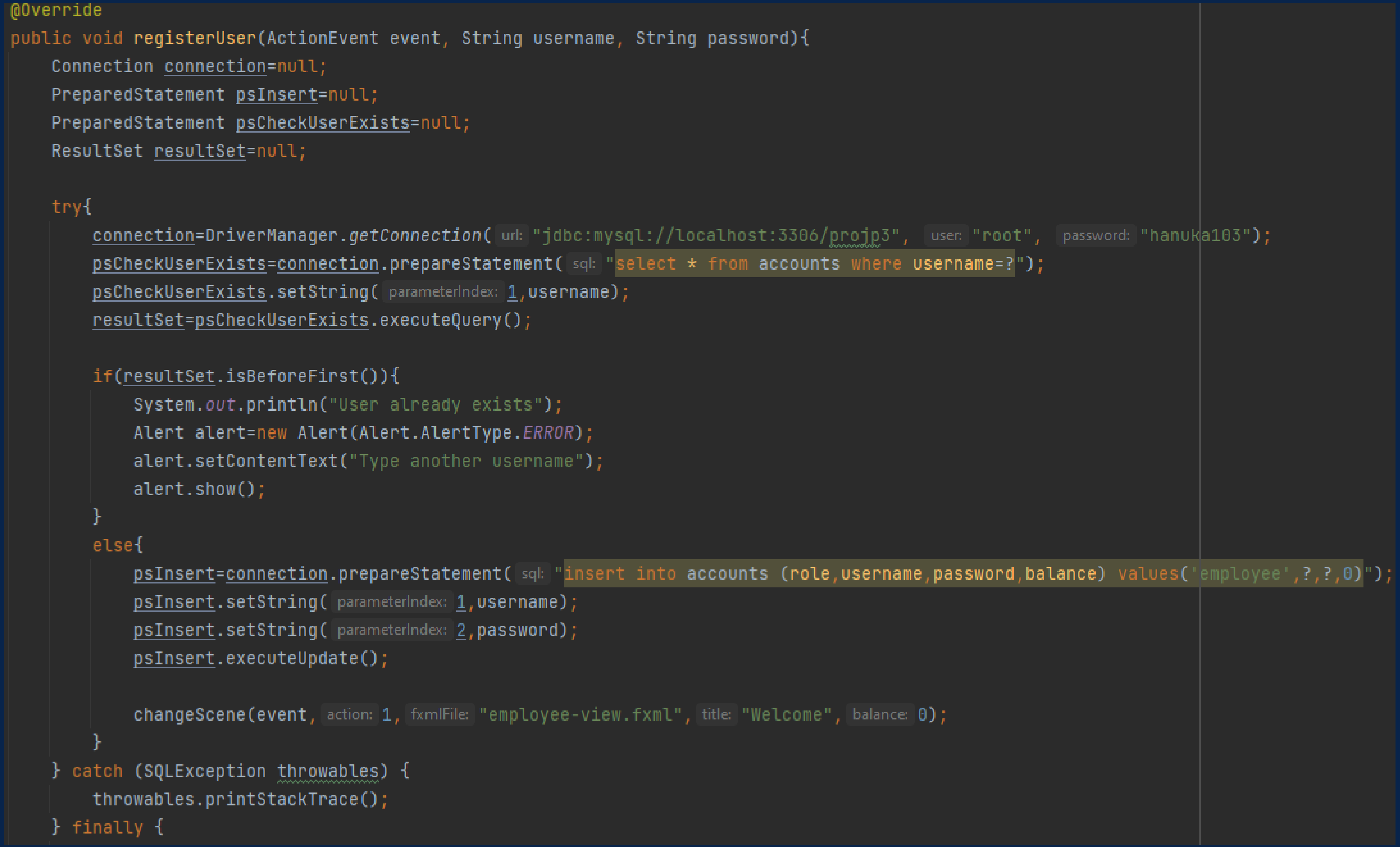


See everything

id	role	user	password	balance
1	client	andrei	123	120.0
2	client	maria	222	1300.0
3	employee	george	111	900.0

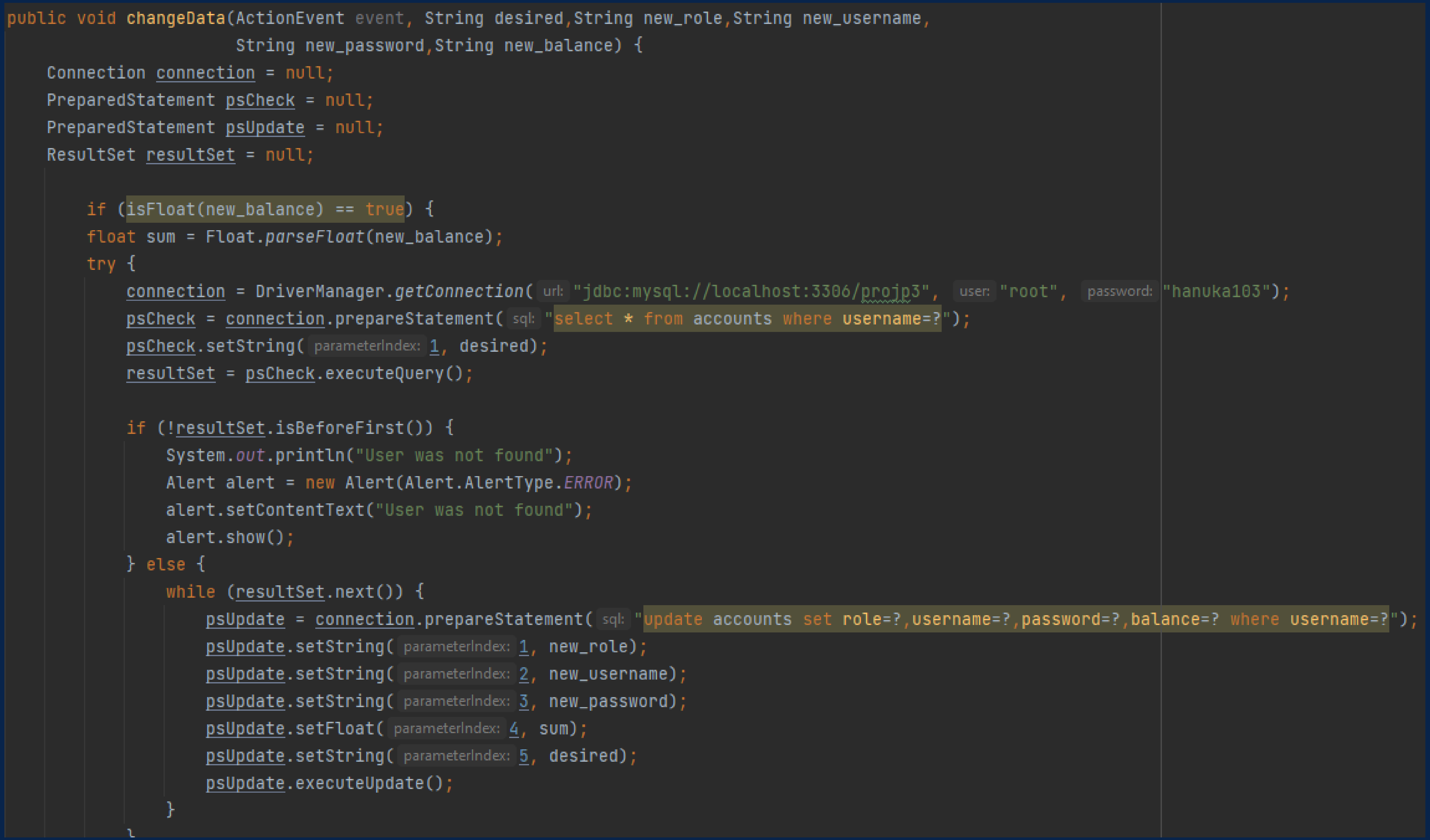
Requirement 3

insert



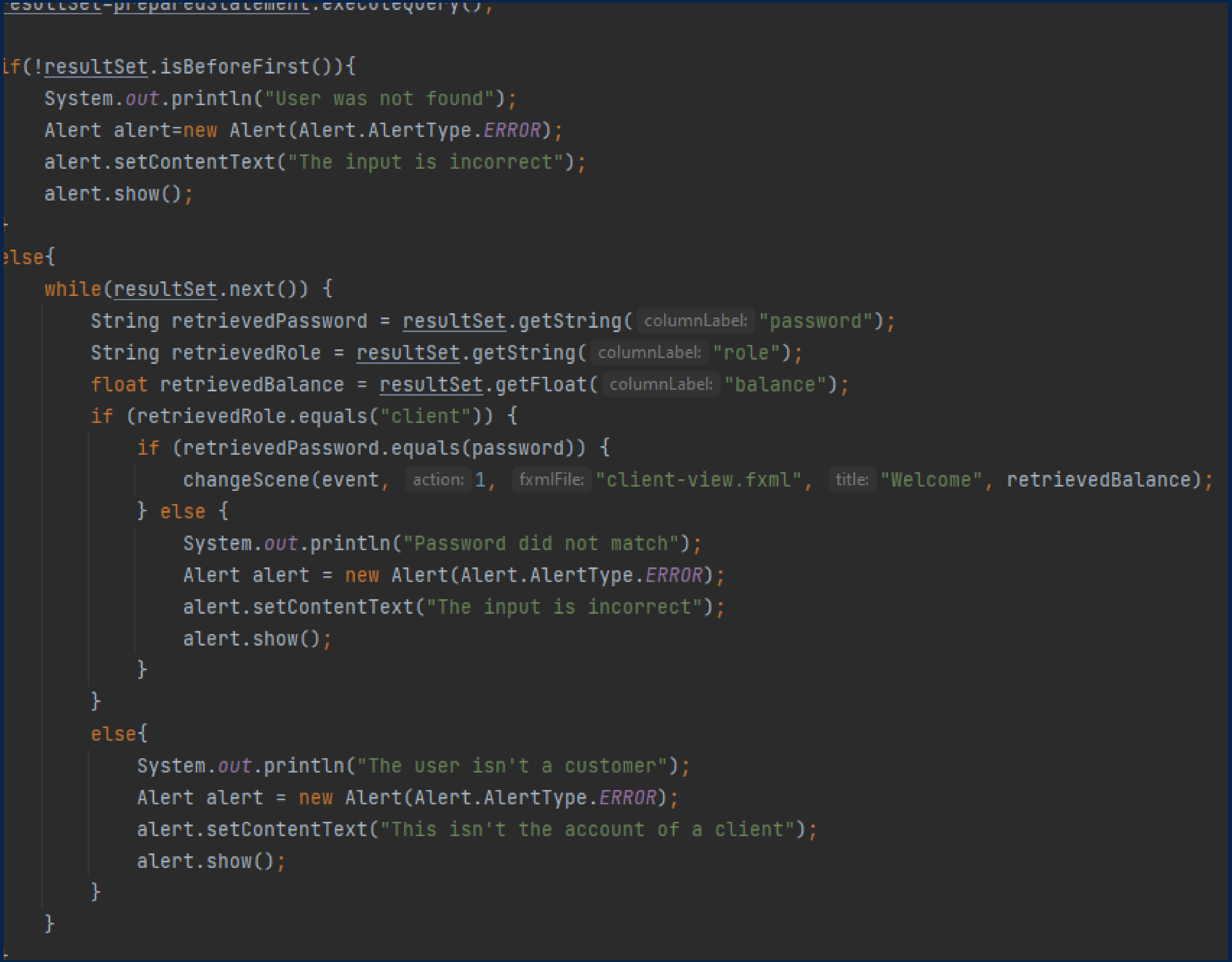
```
@Override
public void registerUser(ActionEvent event, String username, String password){
    Connection connection=null;
    PreparedStatement psInsert=null;
    PreparedStatement psCheckUserExists=null;
    ResultSet resultSet=null;

    try{
        connection=DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/projp3",  user: "root",  password: "hanuka103");
        psCheckUserExists=connection.prepareStatement( sql: "select * from accounts where username=?");
        psCheckUserExists.setString( parameterIndex: 1,username);
        resultSet=psCheckUserExists.executeQuery();

        if(resultSet.isBeforeFirst()){
            System.out.println("User already exists");
            Alert alert=new Alert(Alert.AlertType.ERROR);
            alert.setContentText("Type another username");
            alert.show();
        }
        else{
            psInsert=connection.prepareStatement( sql: "insert into accounts (role,username,password,balance) values('employee',?,?,0)");
            psInsert.setString( parameterIndex: 1,username);
            psInsert.setString( parameterIndex: 2,password);
            psInsert.executeUpdate();

            changeScene(event, action: 1, fxmlFile: "employee-view.fxml", title: "Welcome", balance: 0);
        }
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    } finally {
```

update

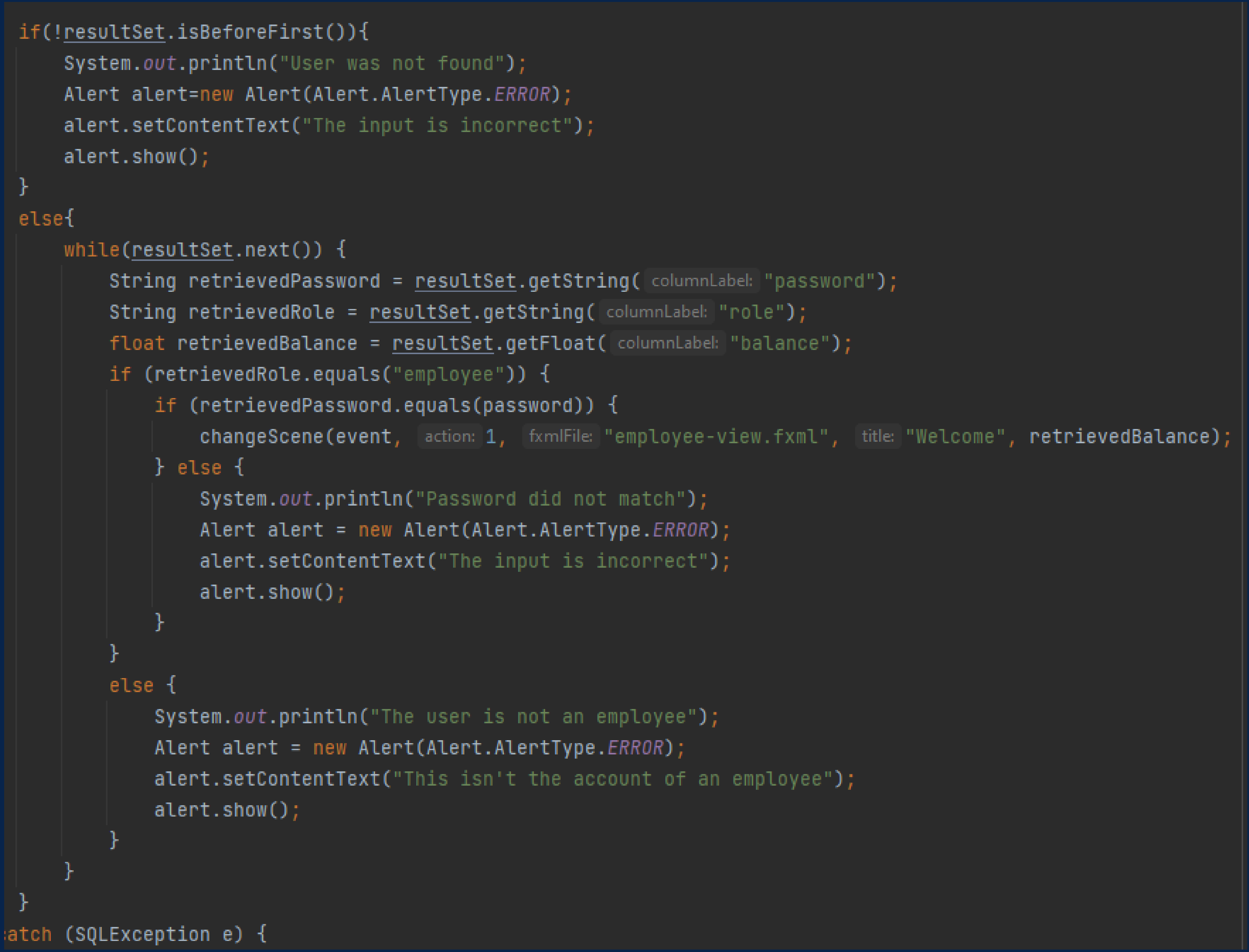


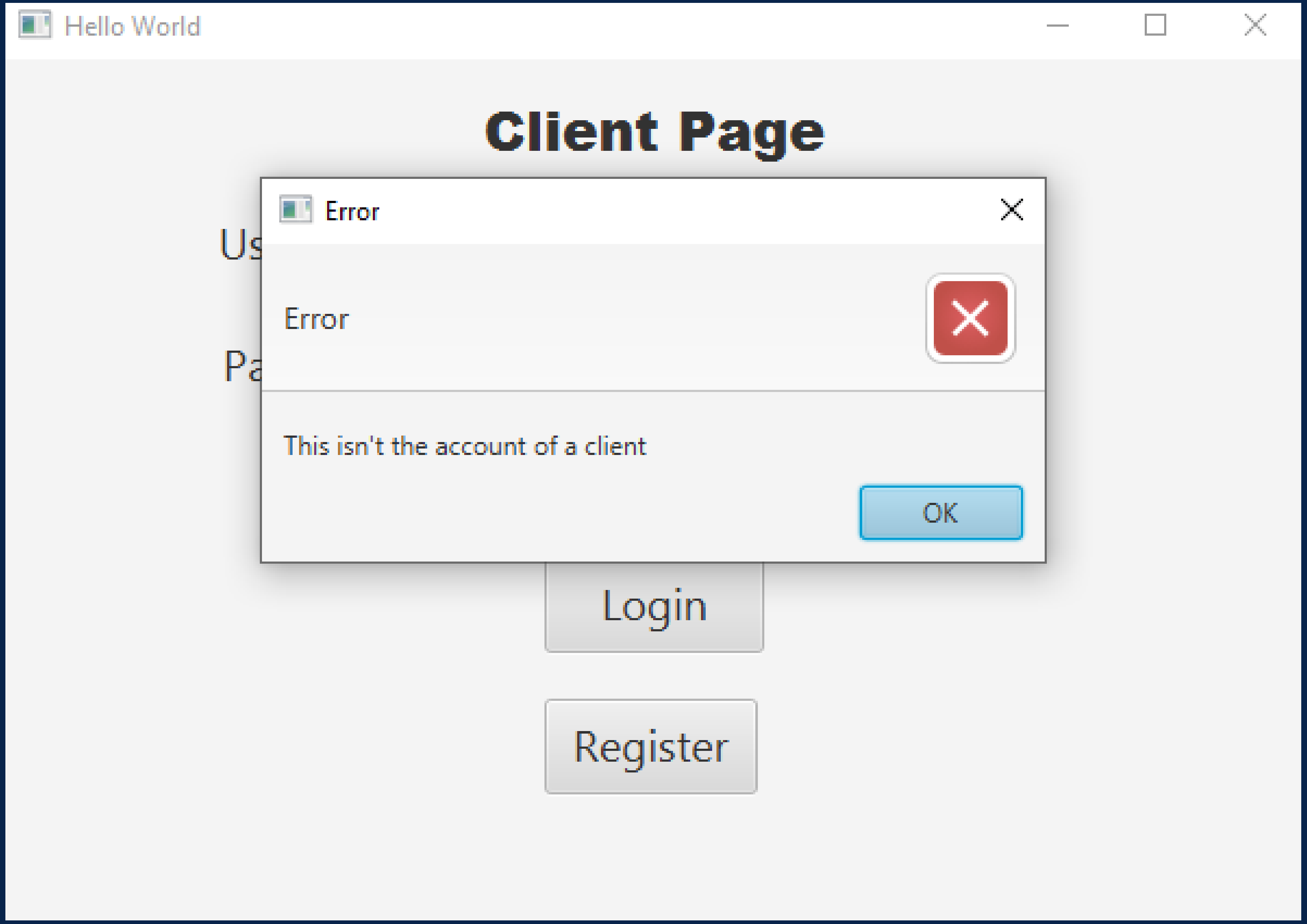
```
public void changeData(ActionEvent event, String desired,String new_role,String new_username,
                       String new_password,String new_balance) {
    Connection connection = null;
    PreparedStatement psCheck = null;
    PreparedStatement psUpdate = null;
    ResultSet resultSet = null;

        if (isFloat(new_balance) == true) {
        float sum = Float.parseFloat(new_balance);
        try {
            connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/projp3",  user: "root",  password: "hanuka103");
            psCheck = connection.prepareStatement( sql: "select * from accounts where username=?");
            psCheck.setString( parameterIndex: 1, desired);
            resultSet = psCheck.executeQuery();

            if (!resultSet.isBeforeFirst()) {
                System.out.println("User was not found");
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setContentText("User was not found");
                alert.show();
            } else {
                while (resultSet.next()) {
                    psUpdate = connection.prepareStatement( sql: "update accounts set role=?,username=?,password=?,balance=? where username=?");
                    psUpdate.setString( parameterIndex: 1, new_role);
                    psUpdate.setString( parameterIndex: 2, new_username);
                    psUpdate.setString( parameterIndex: 3, new_password);
                    psUpdate.setFloat( parameterIndex: 4, sum);
                    psUpdate.setString( parameterIndex: 5, desired);
                    psUpdate.executeUpdate();
                }
```

Requirement 4



```
esultSet=preparedStatement.executeQuery();

if(!resultSet.isBeforeFirst()){
    System.out.println("User was not found");
    Alert alert=new Alert(Alert.AlertType.ERROR);
    alert.setContentText("The input is incorrect");
    alert.show();
}

else{
    while(resultSet.next()) {
        String retrievedPassword = resultSet.getString( columnLabel: "password");
        String retrievedRole = resultSet.getString( columnLabel: "role");
        float retrievedBalance = resultSet.getFloat( columnLabel: "balance");
        if (retrievedRole.equals("client")) {
            if (retrievedPassword.equals(password)) {
                changeScene(event,  action: 1,  fxmlFile: "client-view.fxml",  title: "Welcome", retrievedBalance);
            } else {
                System.out.println("Password did not match");
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setContentText("The input is incorrect");
                alert.show();
            }
        }
        else{
            System.out.println("The user isn't a customer");
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setContentText("This isn't the account of a client");
            alert.show();
        }
    }
}
```


```
if(!resultSet.isBeforeFirst()){
    System.out.println("User was not found");
    Alert alert=new Alert(Alert.AlertType.ERROR);
    alert.setContentText("The input is incorrect");
    alert.show();
}
else{
    while(resultSet.next()) {
        String retrievedPassword = resultSet.getString( columnLabel: "password");
        String retrievedRole = resultSet.getString( columnLabel: "role");
        float retrievedBalance = resultSet.getFloat( columnLabel: "balance");
        if (retrievedRole.equals("employee")) {
            if (retrievedPassword.equals(password)) {
                changeScene(event,  action: 1,  fxmlFile: "employee-view.fxml",  title: "Welcome", retrievedBalance);
            } else {
                System.out.println("Password did not match");
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setContentText("The input is incorrect");
                alert.show();
            }
        }
        else {
            System.out.println("The user is not an employee");
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setContentText("This isn't the account of an employee");
            alert.show();
        }
    }
}
atch (SQLException e) {
```

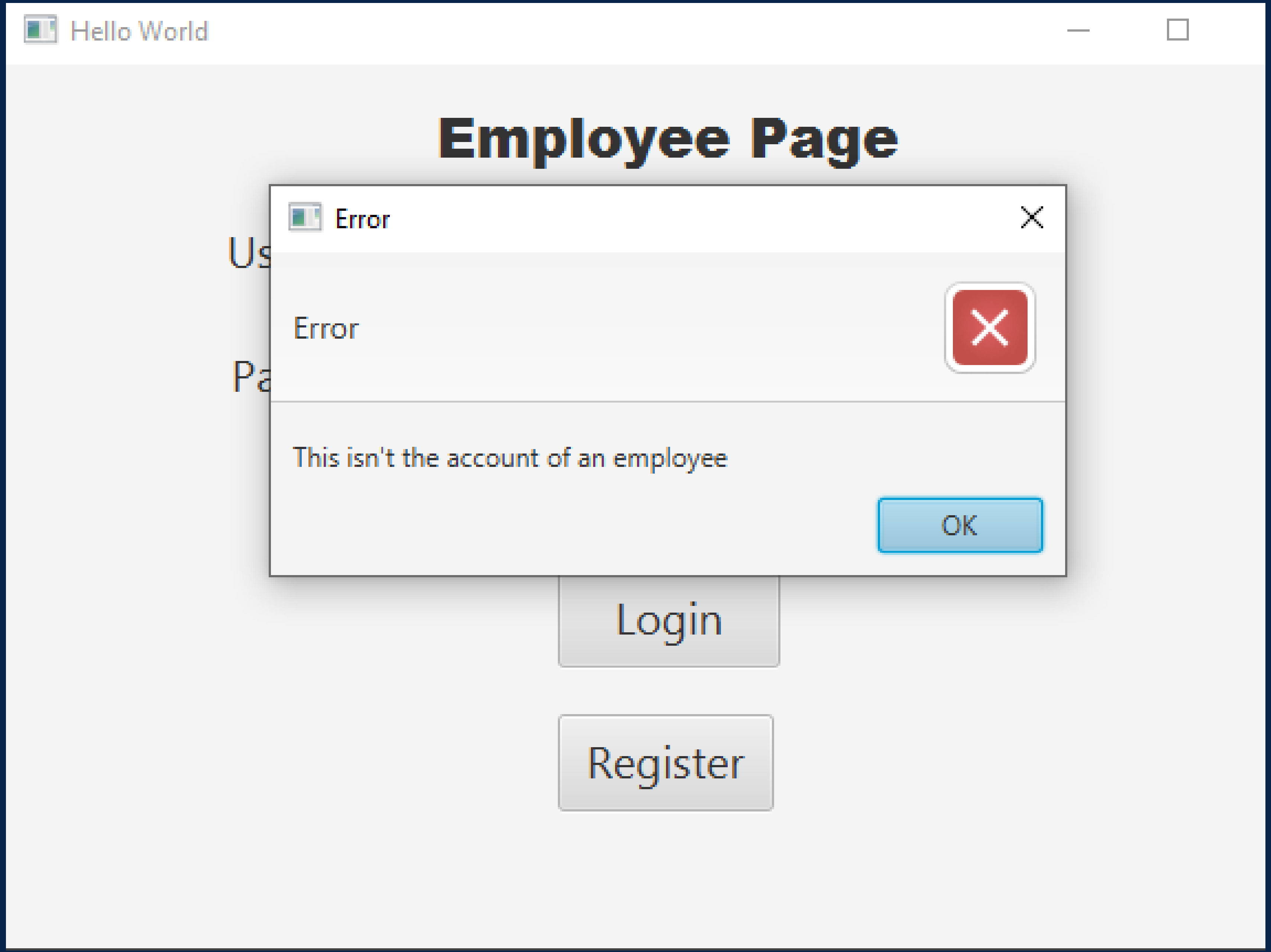

Hello World

Client Page

Error

Error

This isn't the account of a client


Hello World

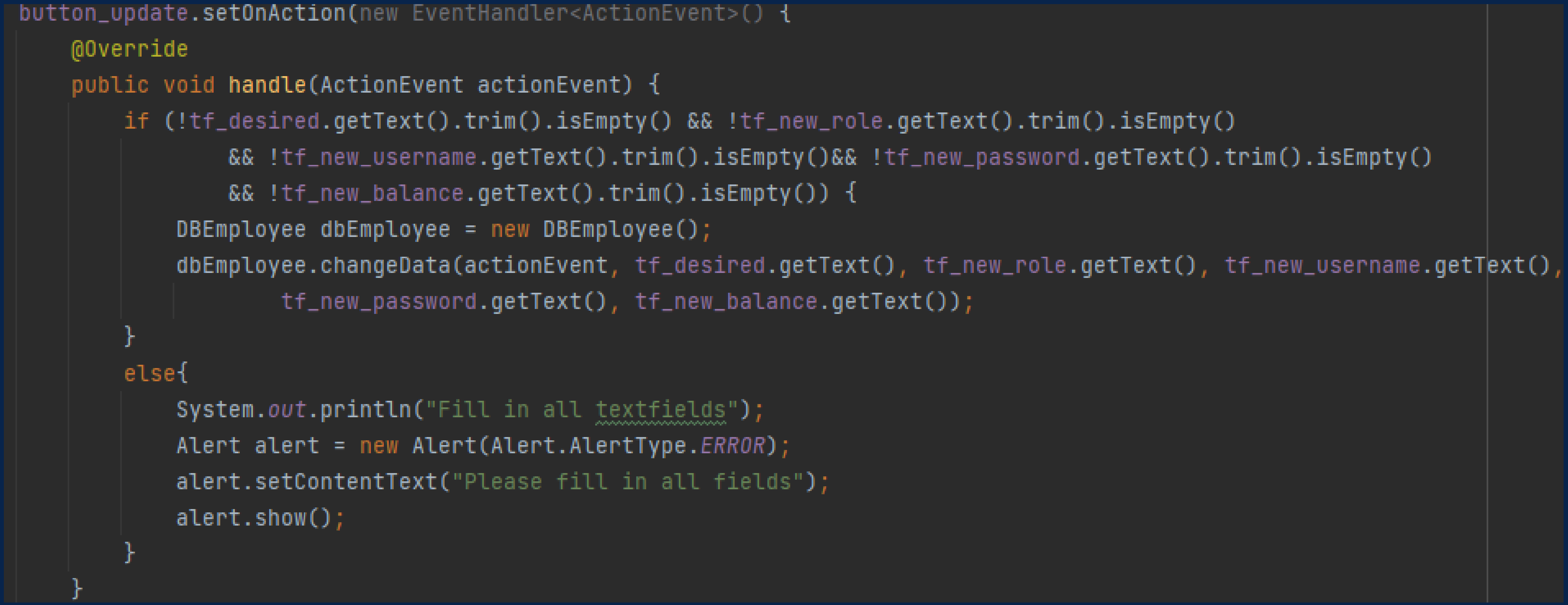
Employee Page

Error

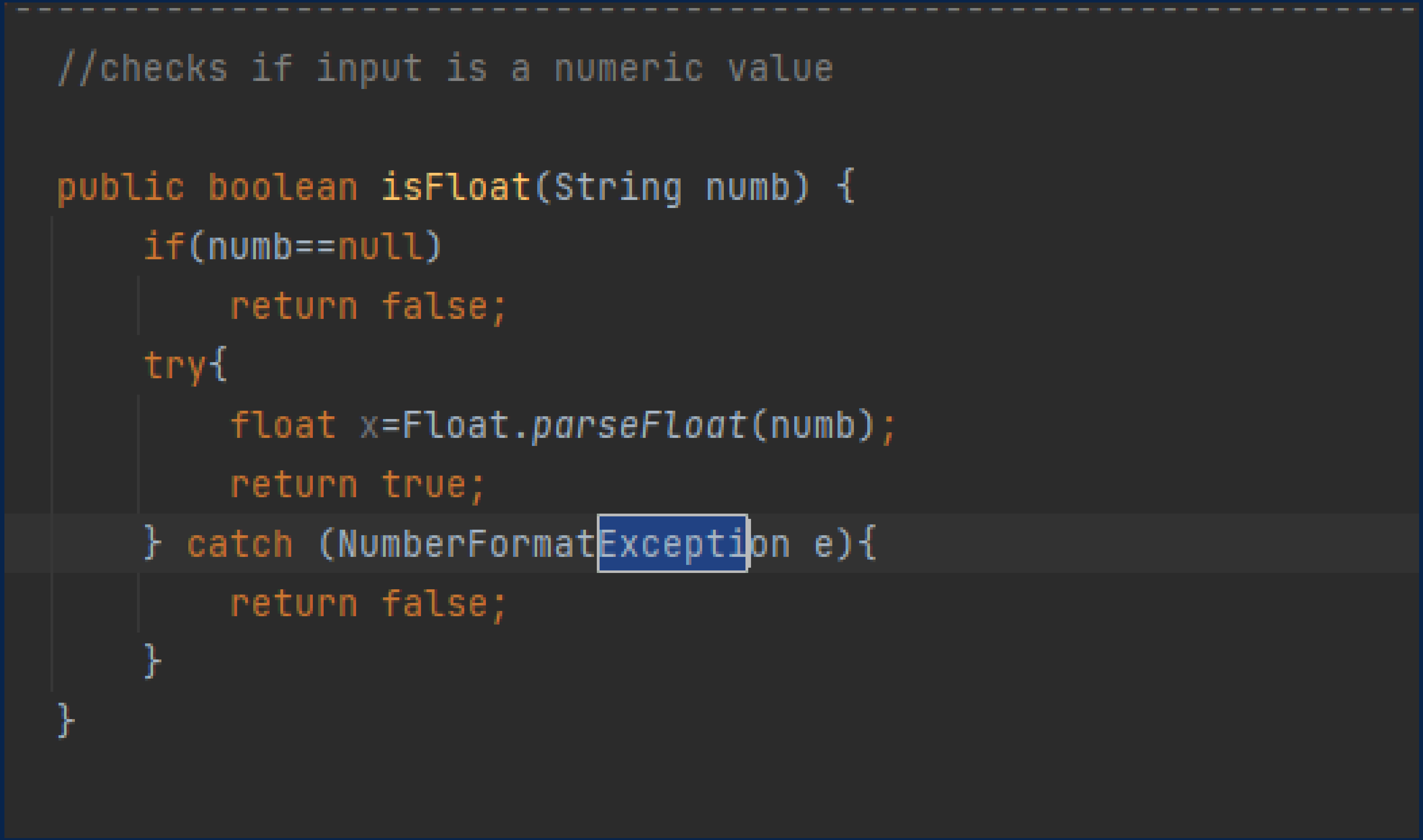
Error

This isn't the account of an employee

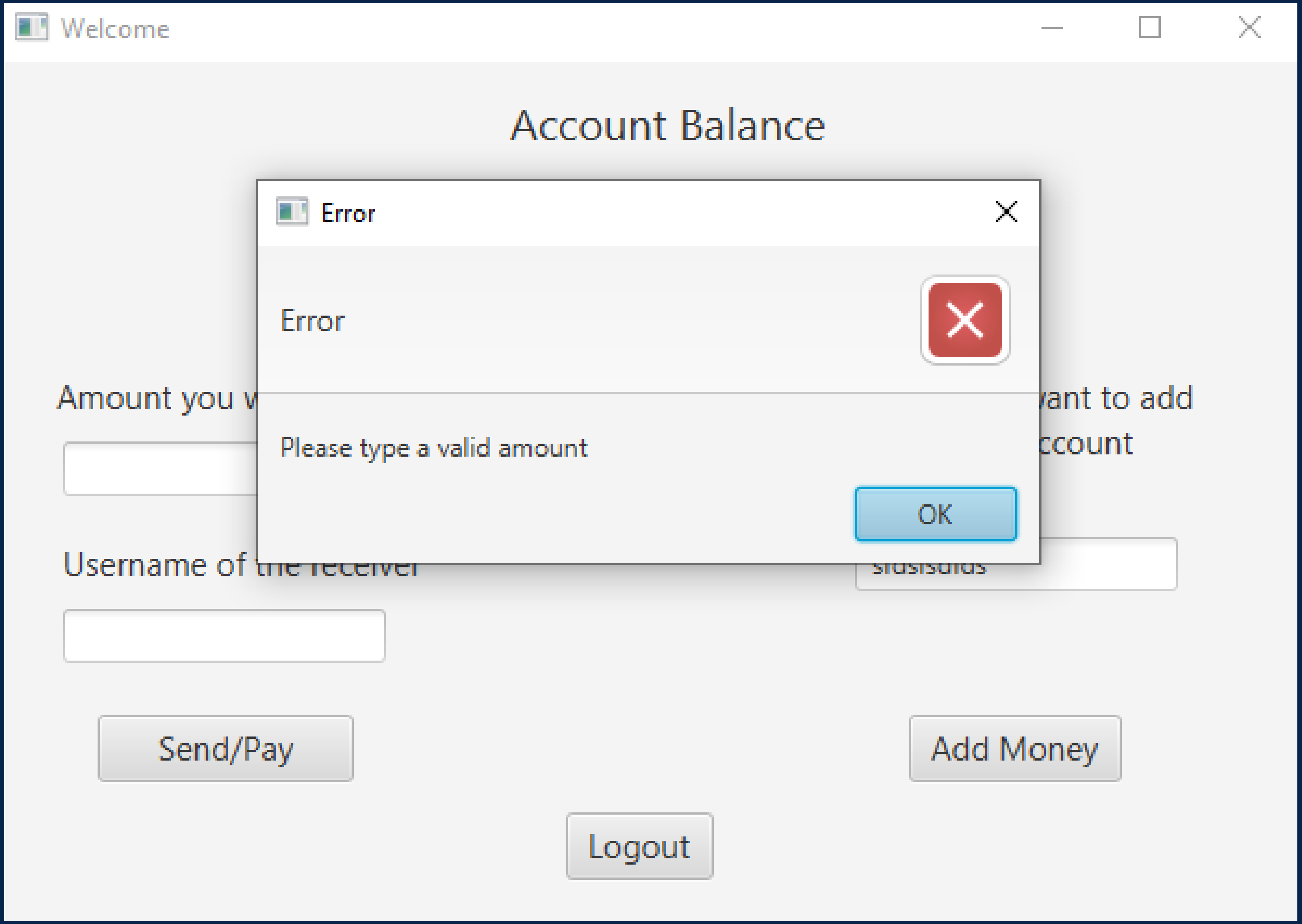
User Interface Requirement 1



```
button_update.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent actionEvent) {
        if (!tf_desired.getText().trim().isEmpty() && !tf_new_role.getText().trim().isEmpty()
                && !tf_new_username.getText().trim().isEmpty()&& !tf_new_password.getText().trim().isEmpty()
                && !tf_new_balance.getText().trim().isEmpty()) {
            DBEmployee dbEmployee = new DBEmployee();
            dbEmployee.changeData(actionEvent, tf_desired.getText(), tf_new_role.getText(), tf_new_username.getText(),
                    tf_new_password.getText(), tf_new_balance.getText());
        }
        else{
            System.out.println("Fill in all textfields");
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setContentText("Please fill in all fields");
            alert.show();
        }
    }
```



```
//checks if input is a numeric value

public boolean isFloat(String numb) {
    if(numb==null)
        return false;
    try{
        float x=Float.parseFloat(numb);
        return true;
    } catch (NumberFormatException e){
        return false;
    }
}
```


Welcome

Account Balance

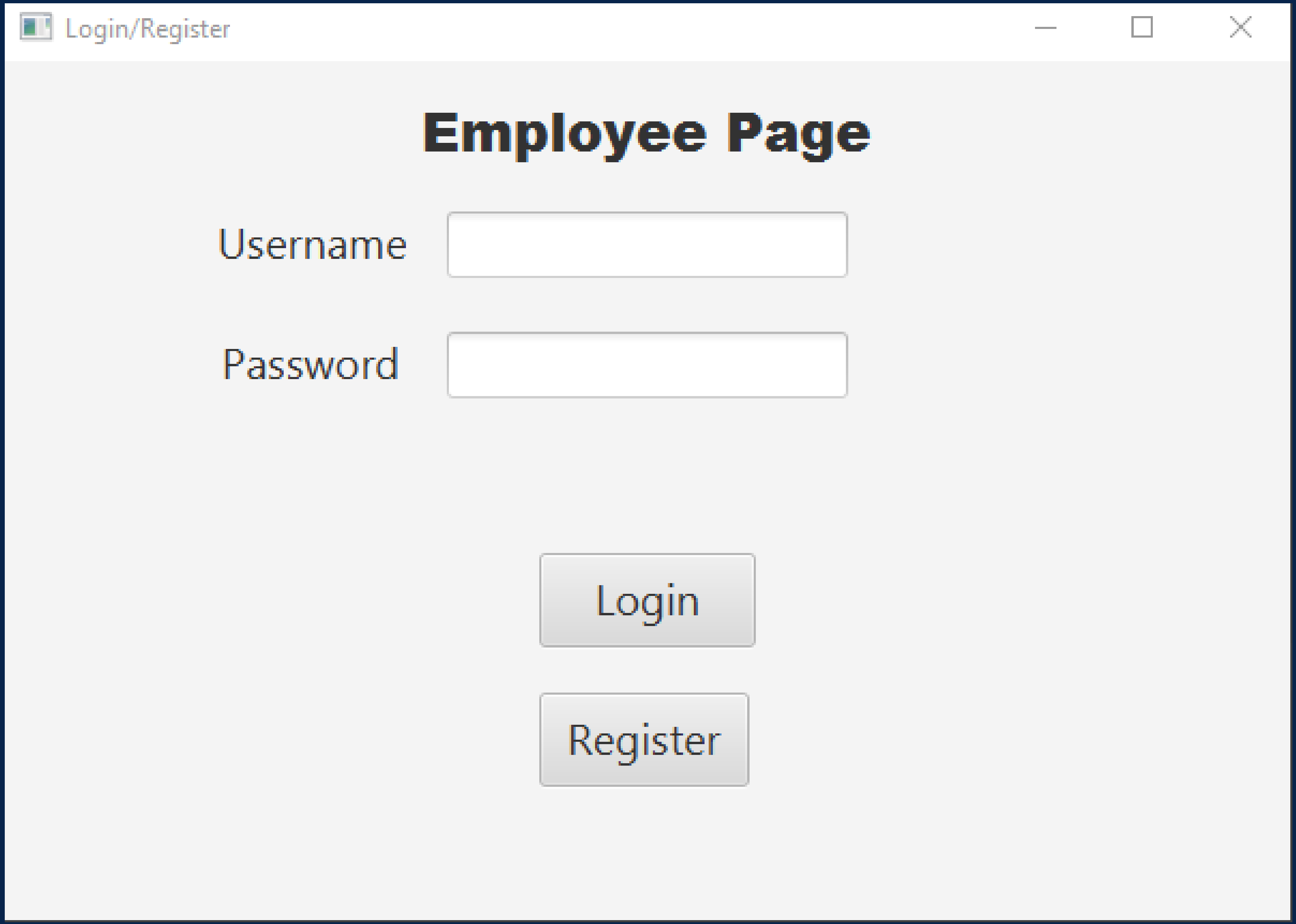
Error

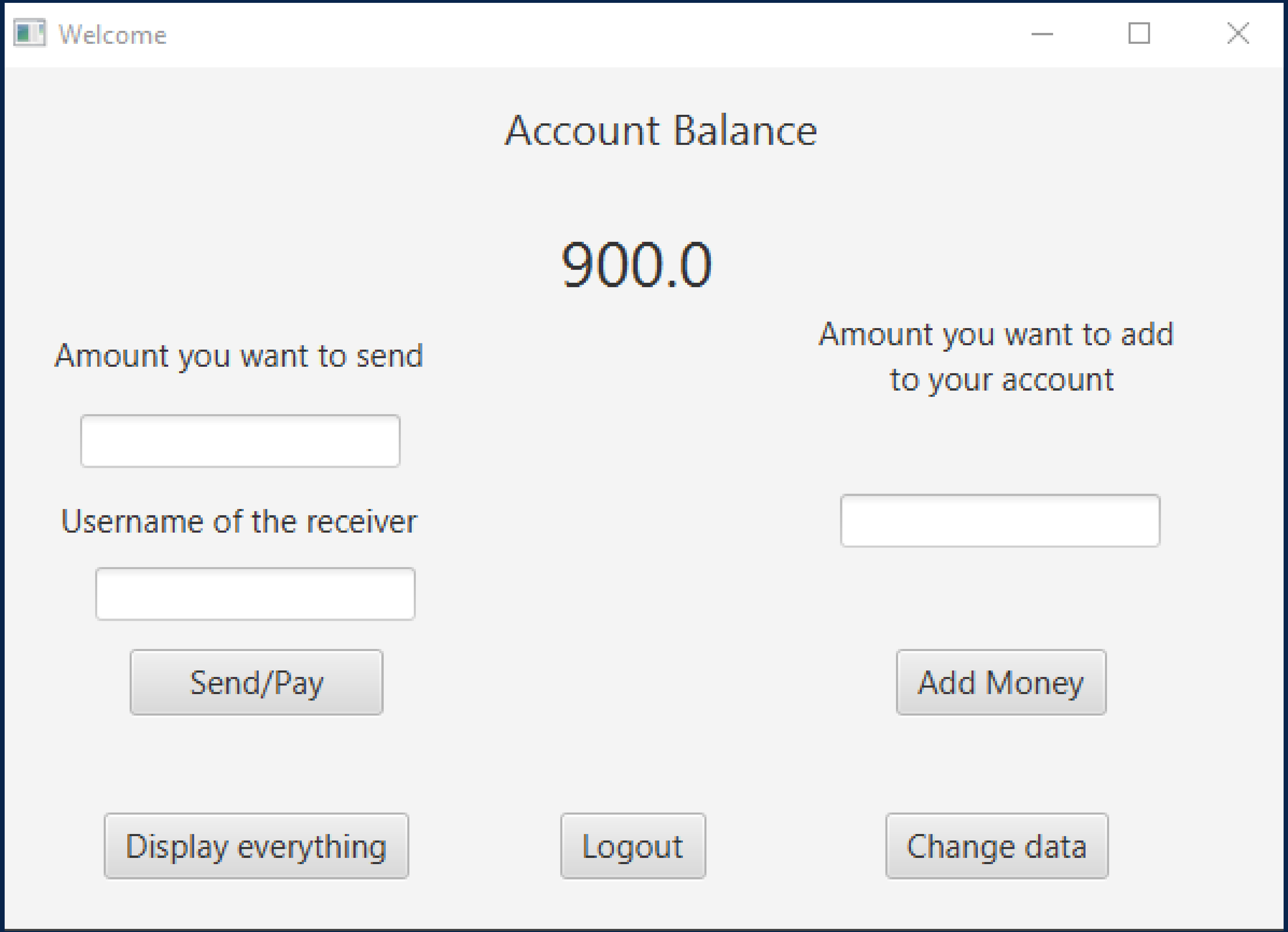
Error

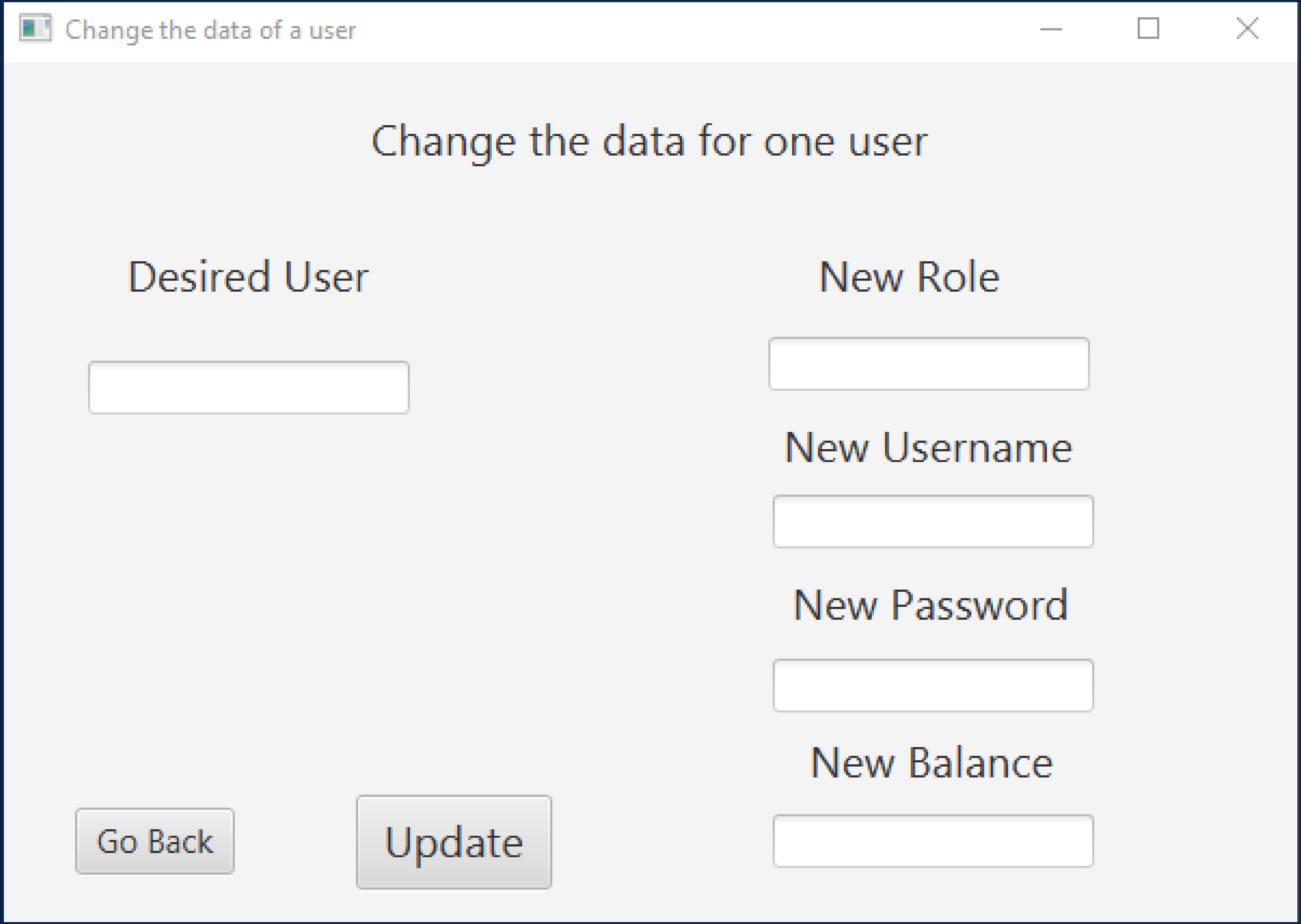
Please type a valid amount

Requirement 2

Employee




Welcome
Account Balance
900.0
Amount you want to send
Username of the receiver
Send/Pay
Amount you want to add to your account
Add Money
Display everything
Logout
Change data


Change the data of a user

Change the data for one user

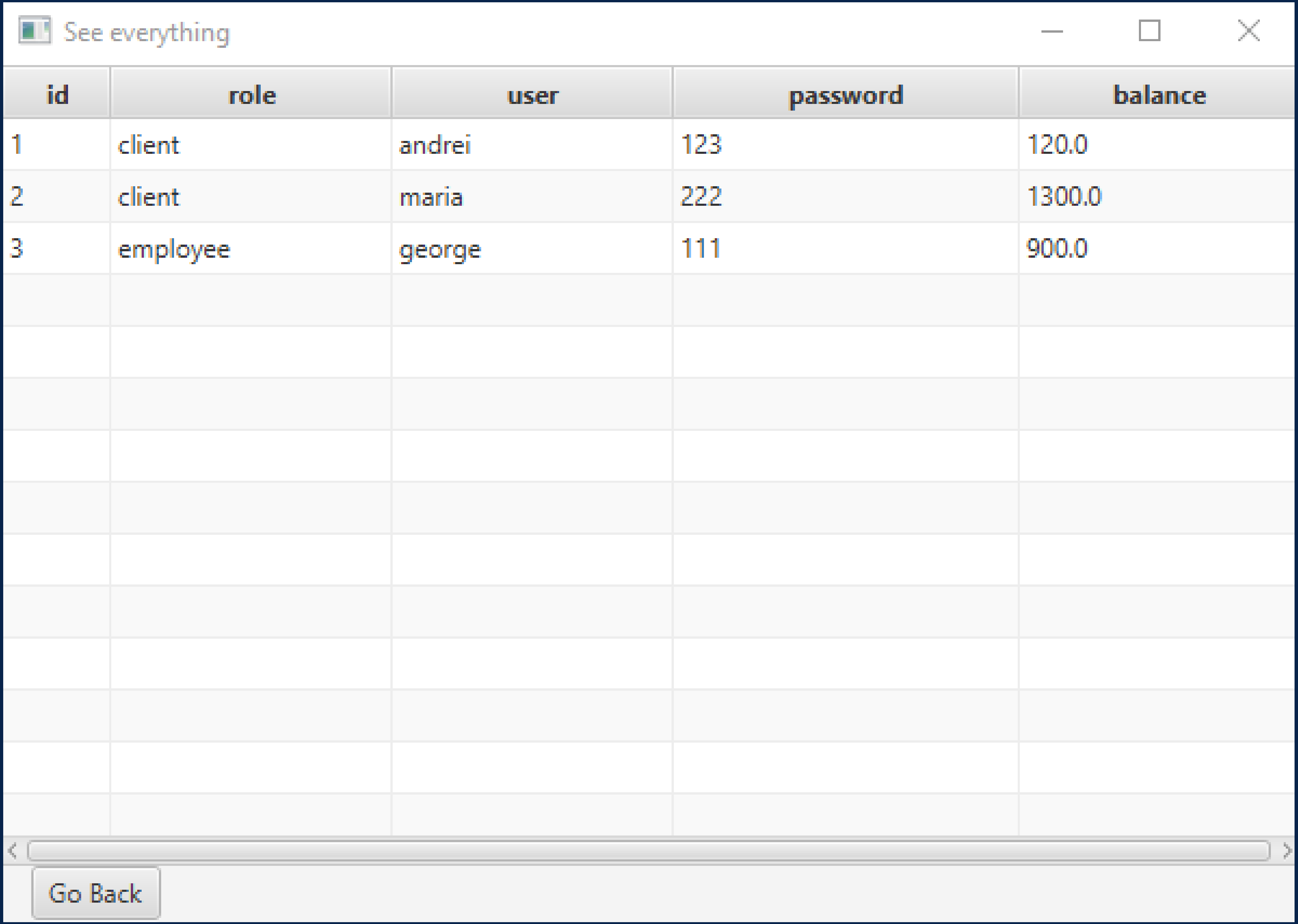
Desired User

New Role

New Username

New Password

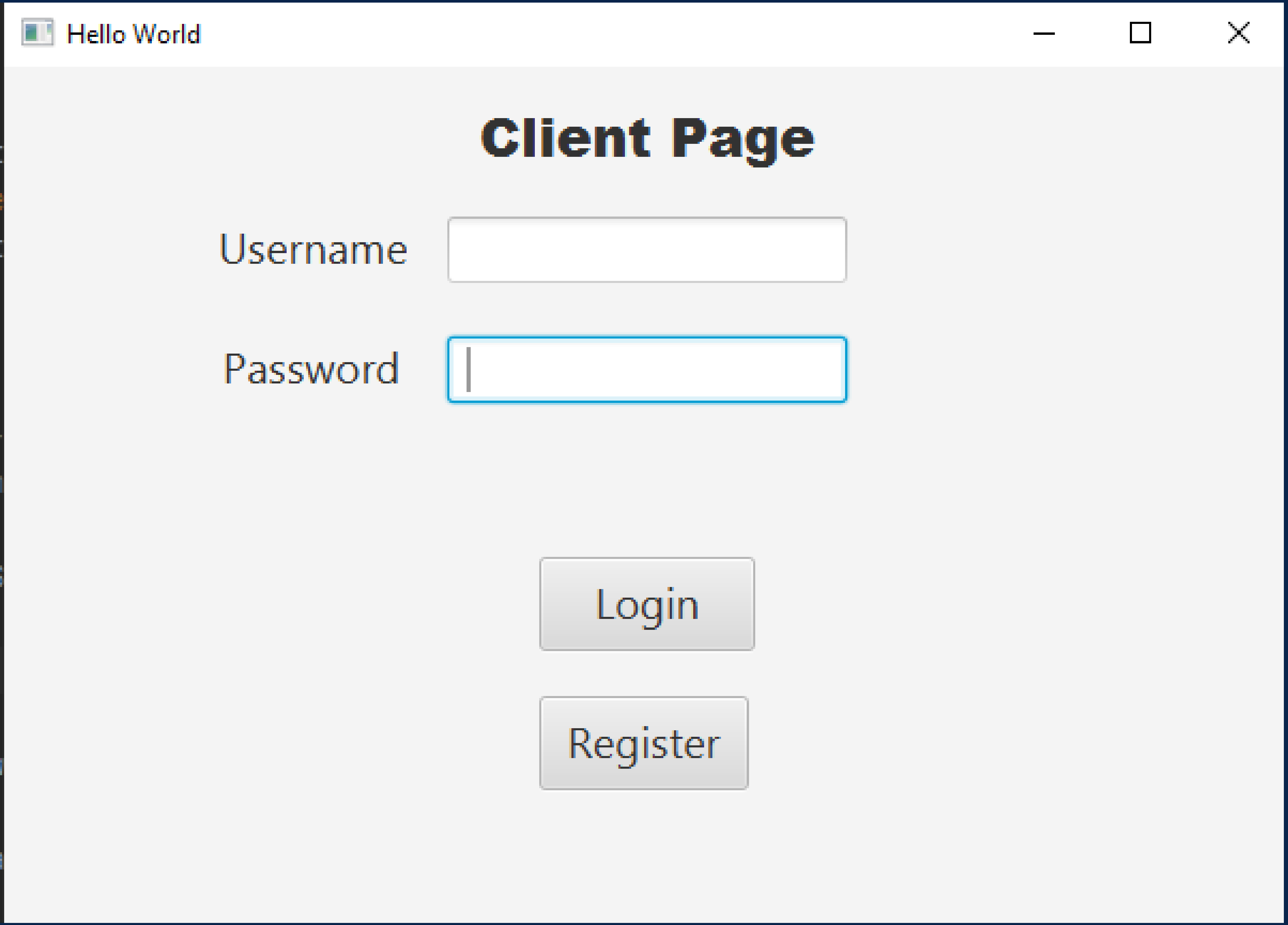
New Balance

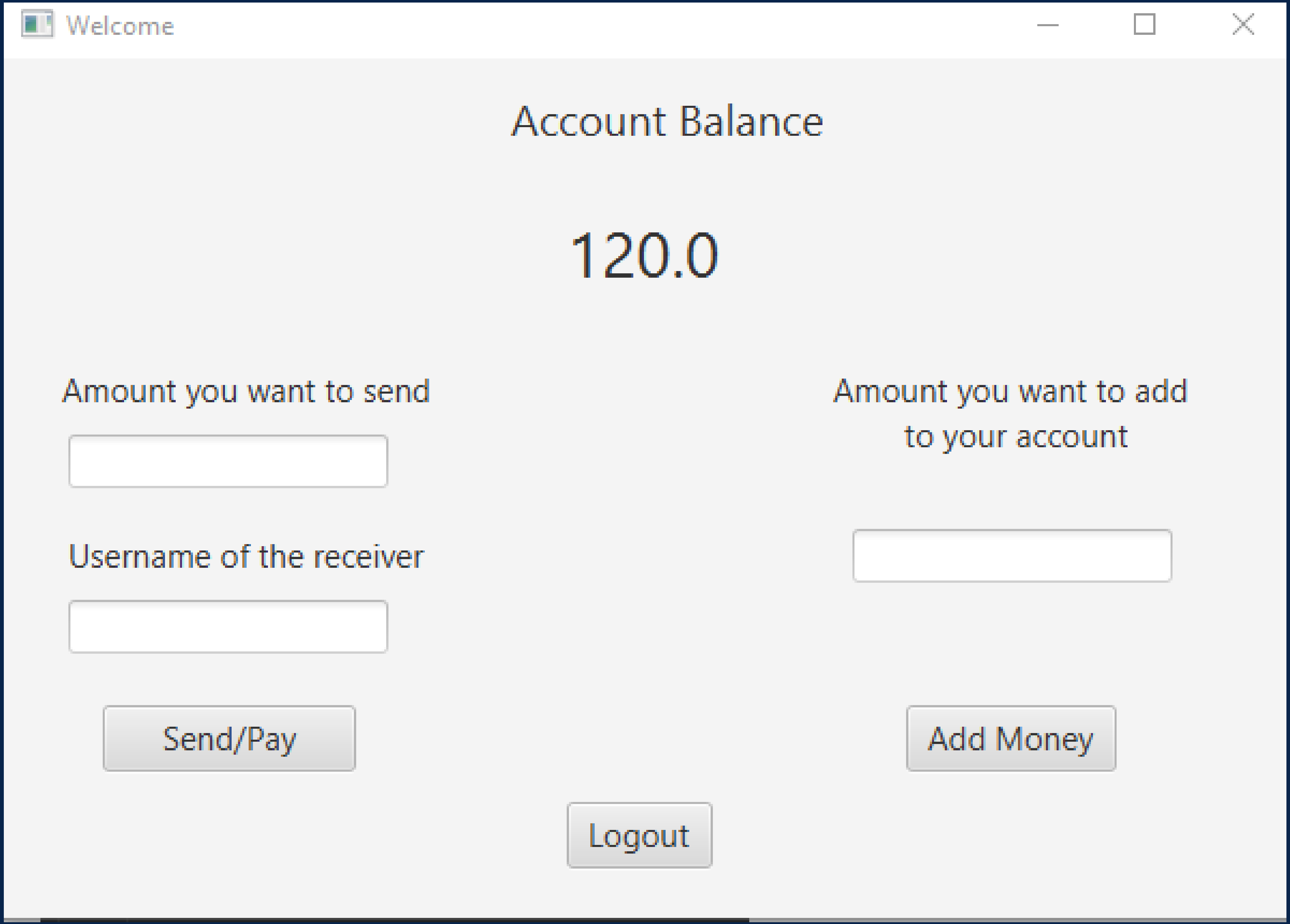


See everything

id	role	user	password	balance
1	client	andrei	123	120.0
2	client	maria	222	1300.0
3	employee	george	111	900.0

Client





Welcome

Account Balance

120.0

Amount you want to send

Username of the receiver

Amount you want to add to your account