Bank System

Wan Yu Yang
903738789
wyang2019@my.fit.edu

Introduction

Background information

Bank is a powerful place for depositing money, so everyone must have at least one bank account. Many companies transfer the salary to employees' bank account instead of give them real money. However, if the companies give their employees real money, the employees do not want to put too much money in their wallet. They want to deposit money to the bank. Therefore, when people want to deposit or withdraw money, they need to go to bank. However, bank tellers are busy every day. They need to deal with many affairs for their customer. It is a routine there is a long line in front of the bank counter. Therefore, if you want to withdraw or deposit money in the counter, you would waste a lot of time. The ATM machine can solve this problem. You can use ATM machine to deposit and withdraw money.

Explain the desired objectives and goals

This project can be used on the ATM machine. People can use their account information to log in bank system. After that, people can deal with some simple thing on the machine. It can decrease the affair of bank tellers. People can save the time in line.

Describe the software that you can realistically provide

People can use this system to withdraw, deposit and transfer money. Through this system, people do not need to spend too much time in line. They could do them

Code

Login: This is an initial page. It starts the program.

```
1  import javafx.application.Application;
6
7  public class Login extends Application {
8      private static Stage stage;
9      public static Stage getStage() { return stage; }
10
11     @Override
12     public void start(Stage stage) throws Exception {
13         Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
14         Scene scene = new Scene(root); // attach scene graph to scene
15         stage.setTitle("Bank System"); // displayed in window's title bar
16         stage.setScene(scene); // attach scene to stage
17         stage.show(); // display the stage
18     }
19
20     public static void main(String[] args) {
21         // create a TipCalculator object and call its start method
22         launch(args);
23     }
24 }
25
```

Login controller:

This class is used to process log in information. If the account number and password are correct, it will enter select type page. Otherwise, it will go back to login page.

```
1 import java.io.IOException;
15
16 public class LoginController {
17
18     //private Customer[] list_customer;
19     private ConnectDB db;
20     private Connection con;
21
22     @FXML
23     private AnchorPane rootPane;
24     @FXML
25     private ResourceBundle resources;
26
27     @FXML
28     private URL location;
29
30     @FXML
31     private TextField input_account;
32
33     @FXML
34     private PasswordField input_password;
35
36     @FXML
37     void clear(ActionEvent event) {
38         input_account.setText("");
39         input_password.setText("");
40     }
```

```
42     @FXML
43     void login(ActionEvent event) throws IOException, SQLException {
44
45         if (db.check_account(con, input_account.getText(), input_password.getText())) {
46             con.close();
47             AnchorPane pane = FXMLLoader.load(getClass().getResource("selectType.fxml"));
48             // System.out.println(pane.getChildren());
49             Label lab = new Label(input_account.getText());
50             lab.setVisible(false);
51             pane.getChildren().add(lab);
52             System.out.println();
53             rootPane.getChildren().setAll(pane);
54         } else {
55             show_alert_message("Error", "Login Fail", "Please check your account number and password again
56             input_account.setText("");
57             input_password.setText("");
58         }
59
60     }
61
62     public void show_alert_message(String title, String header, String detail) {
63         Alert alert = new Alert(AlertType.INFORMATION);
64         alert.setTitle(title);
65         alert.setHeaderText(header);
66         alert.setContentText(detail);
67         alert.showAndWait();
68     }
```

```
69
70      @FXML
71      void initialize() {
72          db = new ConnectDB();
73          try {
74              con = db.getConnection();
75          } catch (SQLException e) {
76              e.printStackTrace();
77          }
78      }
79
80  }
81
```

Select type

This class is used to process selecting type. According to user's option to connect the page. If user selects deposit, it will enter deposit page. If user selects withdraw, it will enter withdraw page. If user selects transfer, it will enter transfer page.

```
1 import java.io.IOException;
11
12 public class SelectTypeController extends LoginController{
13     @FXML
14     private Stage stage;
15
16     public void setStageTitle(String newTitle) {
17         Login.getStage().setTitle(newTitle);
18     }
19
20     @FXML
21     private ResourceBundle resources;
22
23     @FXML
24     private URL location;
25
26     @FXML
27     private AnchorPane rootPane;
28
29
30     @FXML
31     void deposit(ActionEvent event) throws IOException {
32         String log_acc=((Labeled) rootPane.getChildren().get(1)).getText();
33         System.out.println(((Labeled) rootPane.getChildren().get(1)).getText());
34         AnchorPane pane=FXMLLoader.load(getClass().getResource("deposit.fxml"));
35         Label lab=new Label(log_acc);
36         lab.setVisible(false);
```

```
30     @FXML
31     void deposit(ActionEvent event) throws IOException {
32         String log_acc=((Labeled) rootPane.getChildren().get(1)).getText();
33         System.out.println(((Labeled) rootPane.getChildren().get(1)).getText());
34         AnchorPane pane=FXMLLoader.load(getClass().getResource("deposit.fxml"));
35         Label lab=new Label(log_acc);
36         lab.setVisible(false);
37         pane.getChildren().add(lab);
38         rootPane.getChildren().setAll(pane);
39     }
40
41     @FXML
42     void transfer(ActionEvent event) throws IOException {
43         String log_acc=((Labeled) rootPane.getChildren().get(1)).getText();
44         System.out.println(((Labeled) rootPane.getChildren().get(1)).getText());
45         AnchorPane pane=FXMLLoader.load(getClass().getResource("transfer.fxml"));
46         Label lab=new Label(log_acc);
47         lab.setVisible(false);
48         pane.getChildren().add(lab);
49         rootPane.getChildren().setAll(pane);
50     }
51
52     @FXML
53     void withdraw(ActionEvent event) throws IOException {
54         String log_acc=((Labeled) rootPane.getChildren().get(1)).getText();
55         System.out.println(((Labeled) rootPane.getChildren().get(1)).getText());
```

```
41      @FXML
42      void transfer(ActionEvent event) throws IOException {
43          String log_acc=((Labeled) rootPane.getChildren().get(1)).getText();
44          System.out.println(((Labeled) rootPane.getChildren().get(1)).getText());
45          AnchorPane pane=FXMLLoader.load(getClass().getResource("transfer.fxml"));
46          Label lab=new Label(log_acc);
47          lab.setVisible(false);
48          pane.getChildren().add(lab);
49          rootPane.getChildren().setAll(pane);
50      }
51
52      @FXML
53      void withdraw(ActionEvent event) throws IOException {
54          String log_acc=((Labeled) rootPane.getChildren().get(1)).getText();
55          System.out.println(((Labeled) rootPane.getChildren().get(1)).getText());
56          AnchorPane pane=FXMLLoader.load(getClass().getResource("withdraw.fxml"));
57          Label lab=new Label(log_acc);
58          lab.setVisible(false);
59          pane.getChildren().add(lab);
60          rootPane.getChildren().setAll(pane);
61      }
62
63      @FXML
64      void initialize() {
65      }
66  }
67
```

Deposit

This class is used to process depositing. According to user's input, it will update the database.

```
1 import java.net.URL;
12
13 public class DepositController {
14
15     private ConnectDB db;
16     private Connection con;
17
18     @FXML
19     private ResourceBundle resources;
20
21     @FXML
22     private URL location;
23
24     @FXML
25     private AnchorPane rootPane;
26
27     @FXML
28     private TextField input_amount;
29
30     @FXML
31     void submit(ActionEvent event) throws SQLException {
32
33         try {
34             String login_acc = ((Labeled) rootPane.getChildren().get(1)).getText();
35             double balance = db.getbalancce(con, login_acc);
36             double amount = Double.parseDouble(input_amount.getText());
37             System.out.println(input_amount.getText());
```

```
30     @FXML
31     void submit(ActionEvent event) throws SQLException {
32
33         try {
34             String login_acc = ((Labeled) rootPane.getChildren().get(1)).getText();
35             double balance = db.getbalancce(con, login_acc);
36             double amount = Double.parseDouble(input_amount.getText());
37             System.out.println(input_amount.getText());
38             System.out.println(amount + balance);
39
40             db.update(con, login_acc, amount + balance);
41             Alert alert = new Alert(AlertType.INFORMATION);
42             alert.setTitle("Success");
43             alert.setHeaderText("Deposit Success");
44             alert.setContentText("Your balance is "+ (amount + balance));
45             alert.showAndWait();
46         } catch (NumberFormatException e) {
47             Alert alert = new Alert(AlertType.INFORMATION);
48             alert.setTitle("Error");
49             alert.setHeaderText("Deposit Error");
50             alert.setContentText("Please try again.");
51             alert.showAndWait();
52         }
53         con.close();
54         Login.getStage().close();
55     }
```

```
    @FXML
    void initialize() {
        db = new ConnectDB();
        try {
            con = db.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

Withdraw

This class is used to process withdraw. According to user's input, it will update the database. If user wants to withdraw the amount bigger than the balance, it will alert user this transaction is fail and finish the program.

```
1 import java.net.URL;
12
13 public class WithdrawController {
14     private ConnectDB db;
15     private Connection con;
16
17     @FXML
18     private ResourceBundle resources;
19
20     @FXML
21     private URL location;
22
23     @FXML
24     private TextField input_amount;
25
26     @FXML
27     private AnchorPane rootPane;
28
29     @FXML
30     void submit(ActionEvent event) {
31         try {
32             String login_acc = ((Labeled) rootPane.getChildren().get(1)).getText();
33             double balance = db.getbalancce(con, login_acc);
34             double amount = Double.parseDouble(input_amount.getText());
35
36             if (amount > balance) {
37                 Alert alert = new Alert(AlertType.INFORMATION);
```

```
29     @FXML
30     void submit(ActionEvent event) {
31         try {
32             String login_acc = ((Labeled) rootPane.getChildren().get(1)).getText();
33             double balance = db.getbalancce(con, login_acc);
34             double amount = Double.parseDouble(input_amount.getText());
35
36             if (amount > balance) {
37                 Alert alert = new Alert(AlertType.INFORMATION);
38                 alert.setTitle("Error");
39                 alert.setHeaderText("Withdraw error");
40                 alert.setContentText("You do not have enough money.");
41                 alert.showAndWait();
42             } else {
43                 db.update(con, login_acc,  balance-amount);
44                 Alert alert = new Alert(AlertType.INFORMATION);
45                 alert.setTitle("Success");
46                 alert.setHeaderText("Withdraw suceess");
47                 alert.setContentText("Your new balance is "+ db.getbalancce(con, login_acc) );
48                 alert.showAndWait();
49             }
50             con.close();
51             Login.getStage().close();
52         } catch (SQLException e) {
53             e.printStackTrace();
54         }
55     }
```

```
56
57      @FXML
58      void initialize() {
59          db = new ConnectDB();
60          try {
61              con = db.getConnection();
62          } catch (SQLException e) {
63              e.printStackTrace();
64          }
65      }
66  }
67
```

Transfer

This class is used to process transfer. If user input the account is not existed or the transfer amount is bigger than balance, it will alert user the transaction is fail and finish the program. If user input information is correct, it will update the database.

```
1 import java.net.URL;
12
13 public class TransferController {
14
15     private ConnectDB db;
16     private Connection con;
17     @FXML
18     private ResourceBundle resources;
19
20     @FXML
21     private URL location;
22
23     @FXML
24     private AnchorPane rootPane;
25
26     @FXML
27     private TextField tranfer_account;
28
29     @FXML
30     private TextField tranfer_amount;
31
32     @FXML
33     void submit(ActionEvent event) throws SQLException {
34
35         String login_acc = ((Labeled) rootPane.getChildren().get(1)).getText();
36         double balance = db.getbalancce(con, login_acc);
37         double amount = Double.parseDouble(tranfer_amount.getText());
```

```
32     @FXML
33     void submit(ActionEvent event) throws SQLException {
34
35         String login_acc = ((Labeled) rootPane.getChildren().get(1)).getText();
36         double balance = db.getbalancce(con, login_acc);
37         double amount = Double.parseDouble(tranfer_amount.getText());
38
39         String trans_acc=tranfer_account.getText();
40
41         if(db.exist_account(con, trans_acc ) && balance > amount) {
42             double transfer_amount=db.getbalancce(con, trans_acc);
43             db.update(con, login_acc, balance-amount);
44             db.update(con, trans_acc, transfer_amount+amount);
45             Alert alert = new Alert(AlertType.INFORMATION);
46             alert.setTitle("Success");
47             alert.setHeaderText("Transfer Successful!");
48             alert.setContentText("You have transfered to traget account.");
49             alert.showAndWait();
50
51         }else {
52             Alert alert = new Alert(AlertType.INFORMATION);
53             alert.setTitle("Error");
54             alert.setHeaderText("Transfer error");
55             alert.setContentText("Please check information and try it again.");
56             alert.showAndWait();
57         }
58
```

```
50
51          }else {
52              Alert alert = new Alert(AlertType.INFORMATION);
53              alert.setTitle("Error");
54              alert.setHeaderText("Transfer error");
55              alert.setContentText("Please check information and try it again.");
56              alert.showAndWait();
57          }
58
59          con.close();
60          Login.getStage().close();
61      }
62
63      @FXML
64      void initialize() {
65          db = new ConnectDB();
66          try {
67              con = db.getConnection();
68          } catch (SQLException e) {
69              e.printStackTrace();
70          }
71      }
72  }
73
```

ConnectDB

This class is used to interact with database. If the main program needs to query or update the database, it needs to use this class.

```
1  import java.sql.Connection;
9
10
11 public class ConnectDB {
12     public Connection getConnection() throws SQLException {
13         SQLiteConfig config = new SQLiteConfig();
14         // config.setReadOnly(true);
15         config.setSharedCache(true);
16         config.enableRecursiveTriggers(true);
17
18         SQLiteDataSource ds = new SQLiteDataSource(config);
19         ds.setUrl("jdbc:sqlite:Bank.db");
20         return ds.getConnection();
21     }
22
23     // insert data
24     public void insert(Connection con, String acc, String pass, double bal) throws SQLException
25         String sql = "insert into customer (account_number,password,balance) values(?,?,?)";
26         PreparedStatement pst = null;
27         pst = con.prepareStatement(sql);
28         int idx = 1;
29         pst.setString(idx++, acc);
30         pst.setString(idx++, pass);
31         pst.setDouble(idx, bal);
32         pst.executeUpdate();
33     }
```

```
35     public int customer_count(Connection con) throws SQLException {
36         String sql = "select count(*) as row_count from Customer";
37         Statement stat = null;
38         ResultSet rs = null;
39         stat = con.createStatement();
40         rs = stat.executeQuery(sql);
41         return rs.getInt("row_count");
42     }
43
44     public ResultSet selectAll(Connection con) throws SQLException {
45         String sql = "select * from Customer";
46         Statement stat = null;
47         ResultSet rs = null;
48         stat = con.createStatement();
49         rs = stat.executeQuery(sql);
50         return rs;
51     }
```

```
53      public boolean exist_account(Connection con,String acc) throws SQLException {
54          String sql = "select * from Customer where account_number='"+acc+"'";
55          Statement stat = null;
56          ResultSet rs = null;
57          stat = con.createStatement();
58          rs = stat.executeQuery(sql);
59          if (rs.next()) {
60              return true;
61          }
62          else {
63              return false;
64          }
65      }
66
67      public boolean check_account(Connection con, String acc, String pass) throws SQLException {
68          String sql = "select * from Customer where account_number='"+acc+"' and password ='"+pass+"'";
69          Statement stat = null;
70          ResultSet rs = null;
71          stat = con.createStatement();
72          rs = stat.executeQuery(sql);
73          if (rs.next()) {
74              return true;
75          }
76          else {
77              return false;
78          }
79      }


80
81      public double getbalance(Connection con, String acc) throws SQLException {
82          String sql = "select balance  from Customer Where account_number='"+acc+"'";
83          Statement stat = null;
84          ResultSet rs = null;
85          stat = con.createStatement();
86          rs = stat.executeQuery(sql);
87          return rs.getDouble("balance");
88      }
89
90      // update data
91      public void update(Connection con, String acc, double bal) throws SQLException {
92          String sql = "update customer set balance = ? where account_number = ?";
93          PreparedStatement pst = null;
94          pst = con.prepareStatement(sql);
95          int idx = 1;
96          pst.setDouble(idx++, bal);
97          pst.setString(idx++, acc);
98          pst.executeUpdate();
99      }
100
101 }
102
```

Pseudo Code

Login page

enter account number and password
Connect to database to valid the account number and password.
IF account number and password are not correct
    RETURN login page
ELSE
    Enter Select type page

Select Page

IF type is deposit
    Enter Deposit page
Else if type is withdraw
    Enter Withdraw page
Else if type is Transfer
    Enter transfer page

Deposit page

Input amount
Read balance from database
new balance = balance + amount
Write new amount into database
Alert user Deposit is success
Finish the program

Withdraw page

Input amount
Read balance from database
IF amount > balance
    Alert user
ELSE
    new balance = balance – amount
    write new balance into database
    Alert user Withdraw is success
Finish the program

Transfer page

Input transfer account number
Input transfer amount
Read the balance

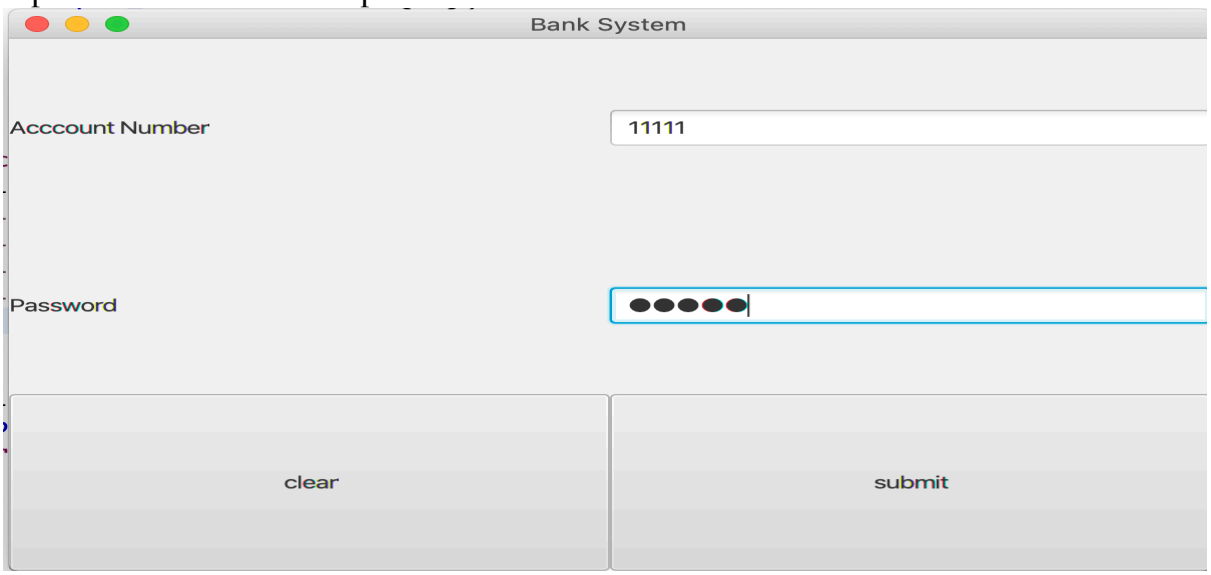
IF account number is invalid or transfer amount > balance
    Alter user

ELSE
    new transfer account balance = transfer account balance+ transfer amount
    new user balance = user balance – transfer amount
    Write new transfer account balance and new user balance into database Alert user
    Alert user transfer is success
Finish the program

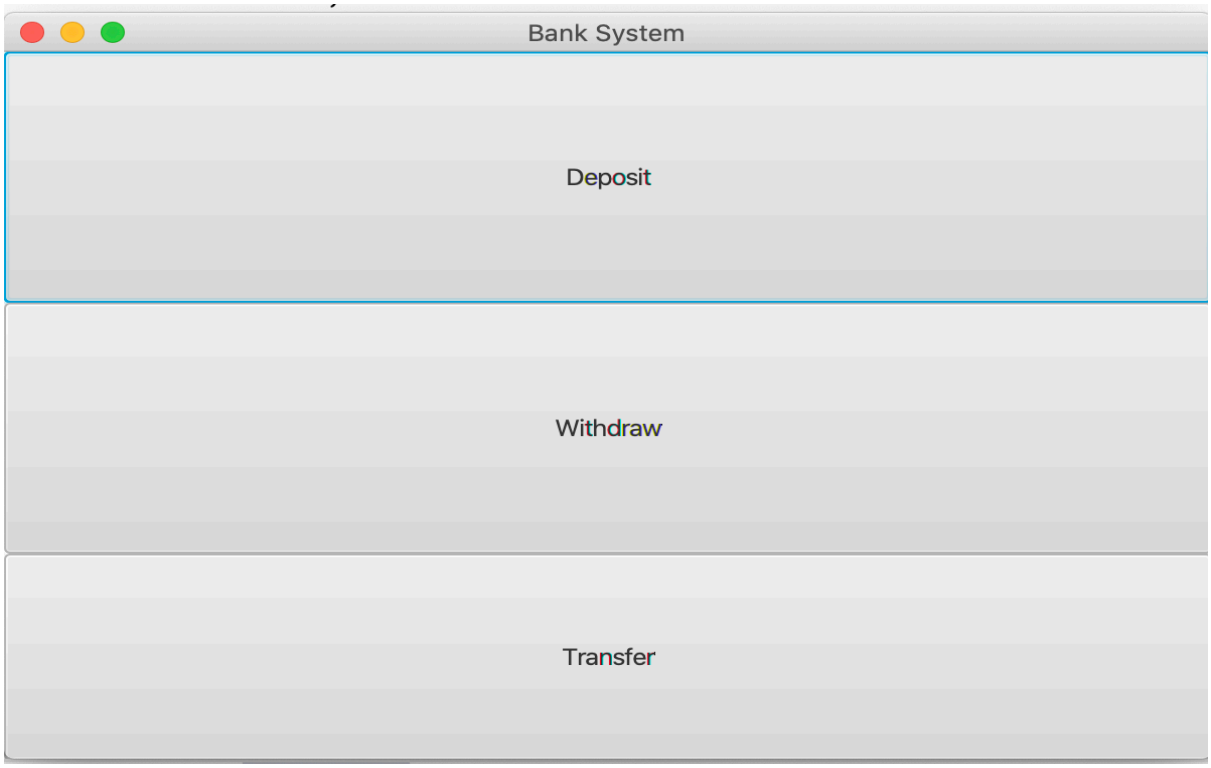
Test

Login

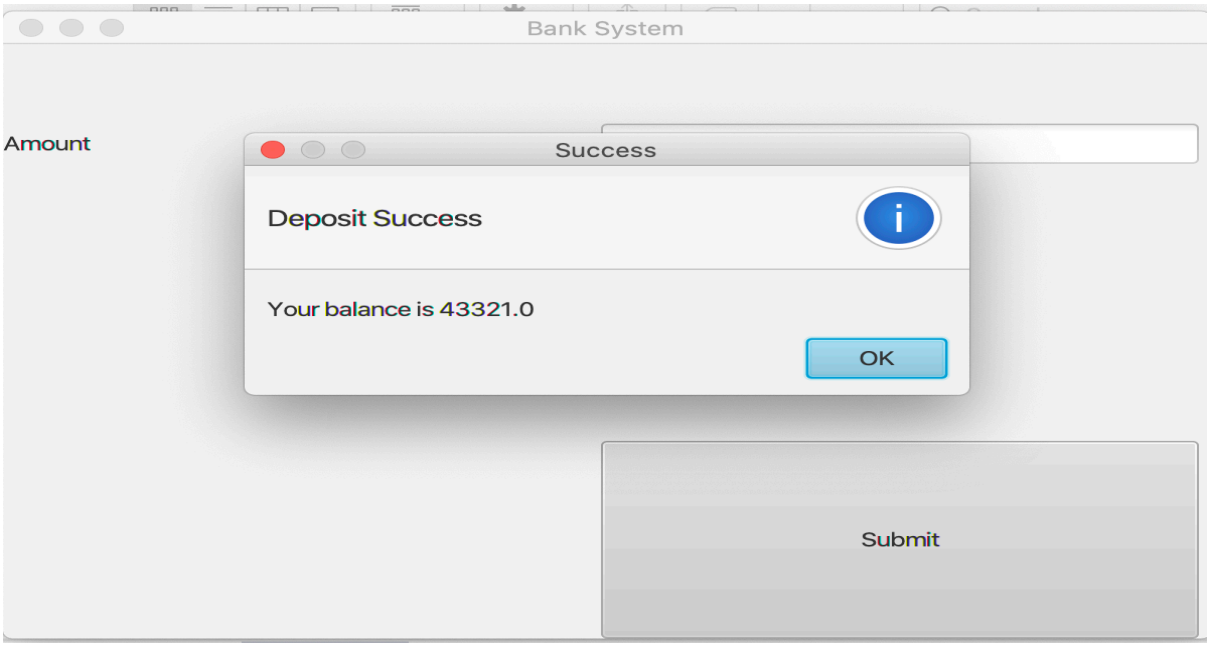
Input account number and password



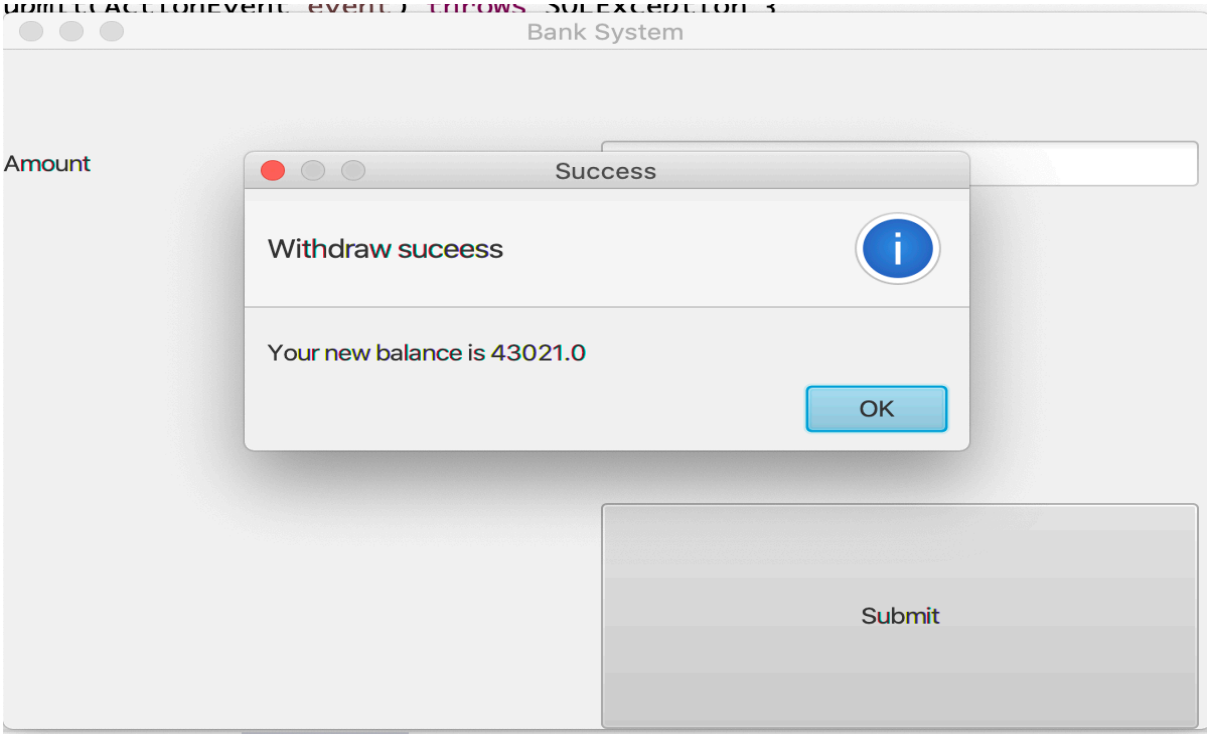
Select type



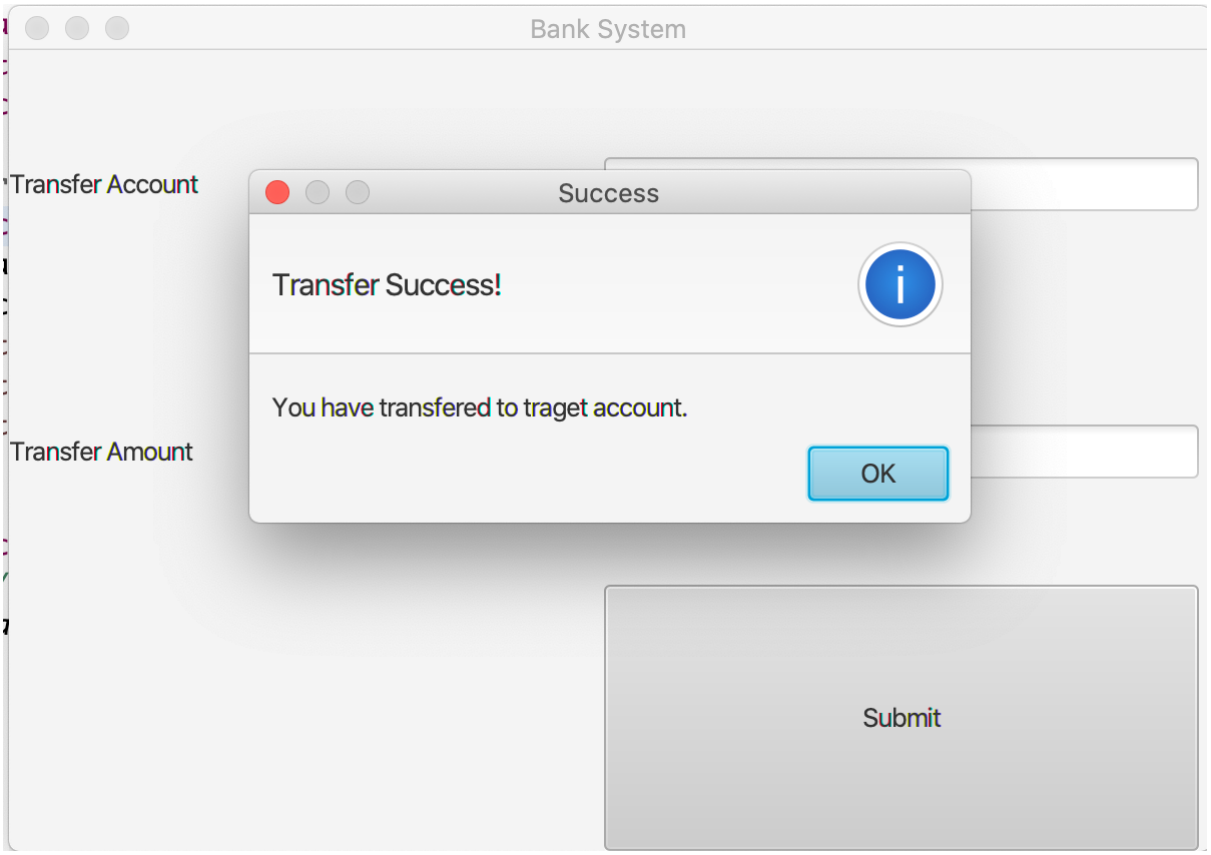
Deposit Success



Withdraw success



Transfer Success



Conclusion

I have completed the project. I learn many skills from this project. I know how to convert page. I learn how to connect database, read data from database and write data into database. It can successfully deposit, withdraw and transfer. I spend a lot of time solving how to transfer a variable to another page. I store a variable in a label. I need to read the content of another page when I want to open another page. Therefore, I create a label and put into the content of next page. I hide this label, so user cannot see this label. When I need to uses this variable, I just need to read the label.

I change my database. At first I want to use MySQL, but I do not have a server to use. Therefore, I use SQLite as my database because it could store in local.

References

[1] Java Database Connectivity Using SQLite: A Tutorial on 4/29/2020
from https://www.researchgate.net/publication/262143238_Java_Database_Connectivity_Using_SQLite_A_Tutorial

[2] SQLite: Light Database System, retrieved on 4/29/2020
from https://ijcsmc.com/docs/papers/April2015/V4I4201599a35.pdf

[3] Java program for banking management system , retrieved on 3/29/2020
from https://www.includehelp.com/java-programs/banking-management-system.aspx

[4] DESIGN AND IMPLEMENTATION OF ONLINE EJB BANK SYSTEM USING JAVA, retrieved on 3/29/2020 from
http://www.ijtre.com/images/scripts/2017040976.pdf