**1. Formulating the Problem**

**1.1 Problem Description**

Design and implement an ATM using Java and JavaFX. The program should allow the user to view their current balance, make a deposit and withdraw money from their account. The user should be able to interact with the program using clickable buttons.

**1.2 Verbalization**

What is the goal?

Display the current balance and deposit/ withdraw money from the account.

What are the givens?

* The Pin Code (Sample value, it can be manually changed)
* The Deposit Amount
* The Withdraw Amount
* Balance (Sample value, it can be manually changed)

What are the unknowns?

* The balance after the deposit or withdraw is executed.

**1.3 Information Elicitation**

Goal

Display the balance , deposit or withdraw money, update the balance.

Givens

* Pin Code
* Balance
* Deposit Amount
* Withdrawal Amount

Unknowns

* Updated balance

Conditions

* User must enter a valid pin in 3 attempts.
* Deposit amount must be less than 2500.
* The withdraw buttons should be accessible only if the balance is larger enough to handle the withdrawal.

**2. Planning the Solution**

**2.1 Solution Strategy**

Get the pin from the user and validate it. If the pin is correct display the current balance and give the user the option to either make a deposit or withdraw money. If the user wants to make a deposit add the deposit amount to the current balance. If the user wants to withdraw money show the withdrawal options to the user depending on the current balance. Once the user has selected a withdraw option subtract the withdraw amount from the current balance. For every transaction display a receipt.

**2.2 Goal Decomposition**

Sub-goal 1

Get the pin from the user.

Sub-goal 2

Validate the pin.

Sub-goal 3

Display the current balance and the options the user has.

Sub-goal 4

If the user selects Deposit, get the amount the user wants to deposit.

If the user selects Withdraw, record the amount the user wants to withdraw based on which withdraw the user has clicked.

Sub-goal 5

If the user has entered a deposit amount, add the amount to the balance.

If the user has selected one of three withdraw option, subtract the amount from the balance.

Sub-goal 6

Display the transaction receipt.

**2.3 Data Organization and Description**

**Input**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Origin** | **Used in Sub-goal #** |
| pin | Pin entered by the user | User | 2 |
| depositAmount | The amount that the user wants to deposit | User | 4 & 5 |
| btnWithdraw50 | Allows the user to withdraw $50 | User | 4 & 5 |
| btnWithdraw100 | Allows the user to withdraw $100 | User | 4 & 5 |
| btnWithdraw200 | Allows the user to withdraw $200 | User | 4 & 5 |

**Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Origin** | **Used in Sub-goal #** |
| Balance | The current balance | Screen | 3 & 6 |

**3. Designing the Solution**

**3.1 Structure Chart**

First Level Decomposition

Goal Refinement

**Sub-goal 1**

Ask the pin from the user.

**Sub-goal 1.1**

Store the pin the user entered.

**Sub-goal 1.2**

Validate the pin.

**Sub-goal 2**

Display the balance and options to the user.

**Sub-goal 3**

Determine what the user wants to do.

**Sub-goal 3.1**

If the user has clicked on the Deposit button, get the amount of deposit from the text field.

**Sub-goal 3.2**

If the user has clicked on one of the Withdraw button, record the amount of withdrawal depending on which withdraw button the user has clicked.

**Sub-goal 4**

Based on the users choice determine whether to make a deposit or withdraw.

**Sub-goal 4.1**

If the active function is Deposit add the deposit amount from Sub-goal 3.1 to the current balance.

**Sub-goal 4.2**

If the active function is Withdraw subtract the withdrawal amount from Sub-goal 3.2 from the current balance.

**Sub-goal 5**

Display the receipt.

Second Level Decomposition

**3.2 Module and Data Specifications**

**Name**: Main - Launch the application.

**Input**: None

**Output**: None

**Logic**: Launch the GUI and get inputs from the user and manipulate the data collected.

**Name**: Start - Create the GUI and use the UI elements to get user input and process the data.

**Input**: Stage object.

**Output**: None

**Logic**: Create the user interface elements and add functionalities to these elements.

**Name**: btnWithdraw50 - Withdraws $50 from the balance.

**Input**: None

**Output**: Balance - 50

**Logic**: Subtract 50 from the current balance.

**Name**: btnWithdraw100 - Withdraws $100 from the balance.

**Input**: None

**Output**: Balance - 100

**Logic**: Subtract 100 from the current balance.

**Name**: btnWithdraw200 - Withdraws $200 from the balance.

**Input**: None

**Output**: Balance - 200

**Logic**: Subtract 200 from the current balance.

**Name**: btnDeposit - Allows the user to make a deposit.

**Input**: Deposit amount.

**Output**: Balance + 200

**Logic**: Adds the deposit amount to the current balance.

**Name**: btnClear - Clears the text field.

**Input**: None

**Output**: Blank text field.

**Logic**: Clears any value that is stored in the text field.

**Name**: btnQuit - Takes the user back to the Enter Pin .

**Input**: None

**Output**: "Instructions Area Please Enter Pin on the numeric keypad."

**Logic**: Set all the values back to their default values and clear the text area.

**Name**: btnEnter - Confirms the user input .

**Input**: Text in the text field.

**Output**: Depends on the active function.

**Logic**: Take the value in the text field and manipulate it according to the active function.

**Data**

|  |  |  |
| --- | --- | --- |
| Name | Type | Structure |
| currentNum | String | **Variable** |
| currentFunction | String | **Variable** |
| password | Integer | **Variable** |
| pin | Integer | **Variable** |
| depositAmount | Integer | **Variable** |
| numTry | Integer | **Variable** |
| Balance | Double | **Variable** |

**3.3 Algorithm**

Logic

**1.** Display a message asking the user to enter the pin.

**1.1** Get the pin from the text field.

**1.2** Validate the pin by comparing it to the default pin.

**2.** Display the total balance and instructions on how to make a deposit or withdraw money.

**3.** Determine what the user wants to do based on the button they click (e.g. Deposit, Withdraw 50, Withdraw 100 or Withdraw 200)

**3.1** If the user clicked Deposit, allow the user to enter the amount they want to deposit and store this input into depositAmount.

**3.2** If the user clicked Withdraw 50 or Withdraw 100 or Withdraw 200 record the withdrawal amount.

**4.** Determine whether to add or subtract from the balance.

**4.1** If the user wants to make a deposit, depositAmount + balance.

**4.2** If the user wants to withdraw money, balance - subtract .

**5.** Display the receipt.

Algorithm Description

First of all the program creates the GUI which allows the user to interact with the program by simply clicking buttons. Each button has an event associated with it. So when the user clicks on a button some action is executed.

The ATM program starts off by displaying the GUI to the screen. The program then asks the user to enter a pin to view the total balance and either make a deposit or withdraw money. Now we need to get the pin that the user entered and compare it to the sample pin. If the pin that the user entered is incorrect display a message informing the user that the pin is incorrect.

Once the pin has been validated, the user is given the option to make a deposit or withdraw money. The user can indicate what they would like to do by either clicking the Deposit button or one of the three Withdraw button.

If the user clicks the Deposit button, allow the user to enter the amount they would like to deposit. If the user enters a value greater than 2500 display a messaging stating that the deposit amount must be less than 2500. If the entered amount is valid, get this amount and store it in a variable. Add the variable containing the deposit amount to the balance.

If the user clicks on one of three withdraw button, subtract the withdraw amount associated with that specific button form the balance. For example, if the user clicks on the Withdraw 50 button subtract 50 from the balance.

After each action print out a receipt that lets the user know that the changes have been saved and their balance has been updated.

**4. Translation**

**4.1 Source Code**

//=======================================================================

// Name : Nidhi Patel

// SID : 31379144

// Course : IT114

// Section : 452

// Instructor : Maura Deek

// Assignment # : Programming Assignment 2

// Date : 03/03/2016

// Description : This program is an ATM(Cash Machine with a simple GUI

//=======================================================================

//import statements

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.stage.Stage;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.ColumnConstraints;

import javafx.geometry.Insets;

import javafx.event.EventHandler;

import javafx.event.ActionEvent;

import java.text.DecimalFormat;

//ATM class that creates the GUI for the ATM and implements the functions of the ATM

public class ATM extends Application {

String currentNum = ""; //currentNum stores the value of the button that is pressed in the keypad

String currentFunction = "Enter Pin"; //currentFunction stores the active function

int password = 1234; //password stores the sample pin

int pin = 0; //pin stores the pin entered by the user

int depositAmount; //depositAmount stores the amount the user has entered to deposit

int numTry = 1; //numTry is a counter to count the number of attemps a user has made to enter the pin

double balance = 1573.91; //balance stores the current balance of the account

//create a currency format to output the currecny values (e.g. balance)

DecimalFormat currency = new DecimalFormat ("$###,###.###");

//main method to launch the program

public static void main(String[]args){

launch(args);

}

//Overridden start method which creats the GUI and manipulates the data used in the program

public void start(Stage primaryStage) throws Exception{

//Set styling properties for the stage

primaryStage.setTitle("Cash Machine");

primaryStage.setResizable(false);

//Create a grid layout

GridPane layout = new GridPane();

//Add styling properties to the stage

layout.setPadding(new Insets(10));

layout.setHgap(5);

layout.setVgap(5);

//Set width for each column in the grid

layout.getColumnConstraints().add(new ColumnConstraints(150));

layout.getColumnConstraints().add(new ColumnConstraints(150));

layout.getColumnConstraints().add(new ColumnConstraints(150));

layout.getColumnConstraints().add(new ColumnConstraints(150));

//Create a text area to display instructions and output to the user

TextArea txtArea = new TextArea();

//Set properties for the text area

txtArea.setPrefColumnCount(50);

txtArea.setPrefRowCount(10);

txtArea.setWrapText(true);

txtArea.setEditable(false);

txtArea.appendText("Instructions Area\nPlease Enter Pin on the numeric keypad.");

//Create a text field to get inputs from the user

TextField inputTxt = new TextField();

//Set propertied for the text field

inputTxt.setEditable(false);

txtArea.setWrapText(true);

//Create buttons

Button btnWithdraw50 = new Button("Withdraw $50");

Button btnWithdraw100 = new Button("Withdraw $100");

Button btnWithdraw200 = new Button("Withdraw $200");

Button btnDeposit = new Button("Deposit");

Button btnQuit = new Button("Quit");

Button btnClear = new Button("Clear");

Button btnEnter = new Button("Enter");

Button btn0 = new Button("0");

Button btn1 = new Button("1");

Button btn2 = new Button("2");

Button btn3 = new Button("3");

Button btn4 = new Button("4");

Button btn5 = new Button("5");

Button btn6 = new Button("6");

Button btn7 = new Button("7");

Button btn8 = new Button("8");

Button btn9 = new Button("9");

//Set styling properties for the buttons

btnWithdraw50.setMaxWidth(Double.MAX\_VALUE);

btnWithdraw50.setDisable(true);

btnWithdraw100.setMaxWidth(Double.MAX\_VALUE);

btnWithdraw100.setDisable(true);

btnWithdraw200.setMaxWidth(Double.MAX\_VALUE);

btnWithdraw200.setDisable(true);

btnDeposit.setMaxWidth(Double.MAX\_VALUE);

btnDeposit.setDisable(true);

btnQuit.setMaxWidth(Double.MAX\_VALUE);

btnClear.setMaxWidth(Double.MAX\_VALUE);

btnEnter.setMaxWidth(Double.MAX\_VALUE);

btn0.setMaxWidth(Double.MAX\_VALUE);

btn1.setMaxWidth(Double.MAX\_VALUE);

btn2.setMaxWidth(Double.MAX\_VALUE);

btn3.setMaxWidth(Double.MAX\_VALUE);

btn4.setMaxWidth(Double.MAX\_VALUE);

btn5.setMaxWidth(Double.MAX\_VALUE);

btn6.setMaxWidth(Double.MAX\_VALUE);

btn7.setMaxWidth(Double.MAX\_VALUE);

btn8.setMaxWidth(Double.MAX\_VALUE);

btn9.setMaxWidth(Double.MAX\_VALUE);

//Add the UI elements to the layout

layout.add(txtArea,0,0,4,1);

layout.add(inputTxt,0,1,4,1);

layout.add(btnWithdraw50,0,2);

layout.add(btn1,1,2);

layout.add(btn2,2,2);

layout.add(btn3,3,2);

layout.add(btnWithdraw100,0,3);

layout.add(btn4,1,3);

layout.add(btn5,2,3);

layout.add(btn6,3,3);

layout.add(btnWithdraw200,0,4);

layout.add(btn7,1,4);

layout.add(btn8,2,4);

layout.add(btn9,3,4);

layout.add(btnDeposit,0,5);

layout.add(btn0,1,5);

layout.add(btnClear,2,5);

layout.add(btnEnter,3,5);

layout.add(btnQuit,0,6);

//Create a scene

Scene scene = new Scene(layout);

//Set the scene onto the stage

primaryStage.setScene(scene);

//Display the stage

primaryStage.show();

//Handle the events related to the 0 button

btn0.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "0";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 1 button

btn1.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "1";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 2 button

btn2.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "2";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 3 button

btn3.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "3";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 4 button

btn4.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "4";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 5 button

btn5.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "5";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 6 button

btn6.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "6";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 7 button

btn7.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "7";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 8 button

btn8.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "8";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the 9 button

btn9.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum += "9";

inputTxt.setText(currentNum);

}

});

//Handle the events related to the Clear button

btnClear.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

inputTxt.setText("");

currentNum = "";

}

});

//Handle the events related to the Quit button

//The user is returned to the Enter Pin screen

btnQuit.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

numTry = 1;

currentFunction = "Enter Pin";

txtArea.setText("");

txtArea.appendText("Instructions Area\nPlease Enter Pin on the numeric keypad.\n");

inputTxt.setText("");

currentNum = "";

pin = 0;

btnWithdraw50.setDisable(true);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

btnDeposit.setDisable(true);

btn0.setDisable(false);

btn1.setDisable(false);

btn2.setDisable(false);

btn3.setDisable(false);

btn4.setDisable(false);

btn5.setDisable(false);

btn6.setDisable(false);

btn7.setDisable(false);

btn8.setDisable(false);

btn9.setDisable(false);

btnEnter.setDisable(false);

}

});

//Handle the events related to the Deposit button

//Activates the Deposit funtion which allows the user to make a deposit

btnDeposit.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

currentNum = "";

inputTxt.setText("");

currentFunction = "Deposit";

txtArea.appendText("\nPlease enter the deposit amount. Maximum deposit possible $2500\n");

btn0.setDisable(false);

btn1.setDisable(false);

btn2.setDisable(false);

btn3.setDisable(false);

btn4.setDisable(false);

btn5.setDisable(false);

btn6.setDisable(false);

btn7.setDisable(false);

btn8.setDisable(false);

btn9.setDisable(false);

btnEnter.setDisable(false);

}

});

//Handle the events related to the Withdraw 50 button

//Allows the user to withdraw $50.00

btnWithdraw50.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

if(balance >= 50){

System.out.println("\nReceipt");

System.out.println("\nCurrent Balance: " + currency.format(balance));

System.out.println("Withdraw Amount: $50.00");

balance -= 50.00;

System.out.println("New Balance: " + currency.format(balance));

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

txtArea.appendText("\nWithdraw Successful \nWithdraw Amount: $50.00\n");

txtArea.appendText("New Balance: " + currency.format(balance)+"\n") ;

inputTxt.setText("");

if(balance < 50)

{

btnWithdraw50.setDisable(true);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

else if(balance < 100)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

else if(balance < 200)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(false);

btnWithdraw200.setDisable(true);

}

}

}

});

//Handle the events related to the Withdraw 100 button

//Allows the user to withdraw $100.00

btnWithdraw100.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

if(balance >= 100){

System.out.println("\nReceipt");

System.out.println("\nCurrent Balance: " + currency.format(balance));

System.out.println("Withdraw Amount: $100.00");

balance -= 100.00;

System.out.println("New Balance: " + currency.format(balance));

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

txtArea.appendText("\nWithdraw Successful \nWithdraw Amount: $100.00\n");

txtArea.appendText("New Balance: " + currency.format(balance)+"\n");

inputTxt.setText("");

if(balance < 50)

{

btnWithdraw50.setDisable(true);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

else if(balance < 100)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

else if(balance < 200 && balance < 100 && balance < 50)

{

btnWithdraw50.setDisable(true);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

}

}

});

//Handle the events related to the Withdraw 200 button

//Allows the user to withdraw $200.00

btnWithdraw200.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

if(balance >= 200){

System.out.println("\nReceipt");

System.out.println("\nCurrent Balance: " + currency.format(balance));

System.out.println("Withdraw Amount: $200.00");

balance -= 200.00;

System.out.println("New Balance: " + currency.format(balance));

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

txtArea.appendText("\nWithdraw Successful \nWithdraw Amount: $200.00\n");

txtArea.appendText("New Balance: " + currency.format(balance) + "\n");

inputTxt.setText("");

if(balance < 50)

{

btnWithdraw50.setDisable(true);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

else if(balance < 100)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(true);

btnWithdraw200.setDisable(true);

}

else if(balance < 200)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(false);

btnWithdraw200.setDisable(true);

}

}

}

});

//Handle the events related to the Enter button

//Allows the user to send the input to the program

btnEnter.setOnAction(new EventHandler<ActionEvent>(){

public void handle(ActionEvent event){

//If the active funtion is Enter Pin

//Check if the user has entered the correct pin

//Check if the user has entered a blank value

//Check if the user has entered the wrong pin

//Exit the program if the user has 3 incorrect attempts

if(currentFunction == "Enter Pin")

{

if((inputTxt.getText().isEmpty()))

{

txtArea.appendText("\nPlease Enter Pin on the numeric keypad.\n");

}

else

{

pin = Integer.parseInt(inputTxt.getText());

}

if(pin == password)

{

txtArea.appendText("\nTotal Balance: " + currency.format(balance) + "\nTo make a deposit click the Deposit button.\nTo withdraw click on the Withdraw button.\n");

inputTxt.setText("");

currentNum = "";

btn0.setDisable(true);

btn1.setDisable(true);

btn2.setDisable(true);

btn3.setDisable(true);

btn4.setDisable(true);

btn5.setDisable(true);

btn6.setDisable(true);

btn7.setDisable(true);

btn8.setDisable(true);

btn9.setDisable(true);

btnEnter.setDisable(true);

if(balance >= 50)

{

btnWithdraw50.setDisable(false);

}

if (balance >= 100)

{

btnWithdraw100.setDisable(false);

}

if(balance >= 200)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(false);

btnWithdraw200.setDisable(false);

}

btnDeposit.setDisable(false);

}

else if (pin != password && numTry < 3 && (!inputTxt.getText().isEmpty()))

{

txtArea.appendText("\nIncorrect Pin Try Again.\n");

numTry++;

}

else if (numTry == 3)

{

primaryStage.close();

}

}

//If the active funtion is Deposit

//Check if the user has entered a black value

//Check if the user has entered a value greater than 2500

//If the user has entered a valid value add the deposit amount

//to the current balnce and display the transaction in the text area

else if (currentFunction == "Deposit")

{

if((inputTxt.getText().isEmpty()))

{

txtArea.appendText("\nPlease Enter Deposit Amount.\n");

}

else

{

depositAmount = Integer.parseInt(inputTxt.getText());

if(depositAmount >= 2500)

{

txtArea.appendText("\nDeposit Amount Must Be Less Than $2500.00\n");

}

else if (depositAmount < 2500 )

{

System.out.println("\nReceipt");

System.out.println("Current Balance: " + currency.format(balance));

System.out.println("Deposit Amount: " + currency.format(depositAmount));

balance += depositAmount;

System.out.println("New Balance: " + currency.format(balance));

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

txtArea.appendText("\nDeposit Successful \nDeposit Amount: $"+ depositAmount + "\n");

txtArea.appendText("New Balance: " + currency.format(balance) + "\n") ;

inputTxt.setText("");

depositAmount = 0;

currentNum = "";

btn0.setDisable(true);

btn1.setDisable(true);

btn2.setDisable(true);

btn3.setDisable(true);

btn4.setDisable(true);

btn5.setDisable(true);

btn6.setDisable(true);

btn7.setDisable(true);

btn8.setDisable(true);

btn9.setDisable(true);

btnEnter.setDisable(true);

if(balance >= 50)

{

btnWithdraw50.setDisable(false);

}

if (balance >= 100)

{

btnWithdraw100.setDisable(false);

}

if(balance >= 200)

{

btnWithdraw50.setDisable(false);

btnWithdraw100.setDisable(false);

btnWithdraw200.setDisable(false);

}

}

}

}

}

});

}

}

**4.2 Program and Module Description**

Main

Launch the application.

Start

Create a GUI for the ATM and manipulate the balance according to the transactions made.

**5. Solution Testing**

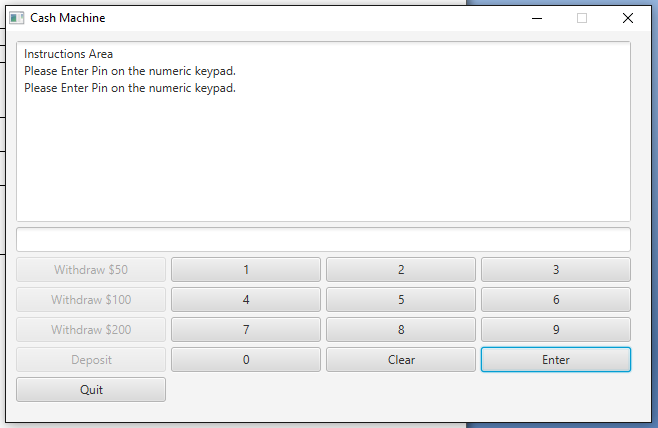
|  |  |  |  |
| --- | --- | --- | --- |
| **ATM Pin Testing** | | | |
| **Test Case #** | **Pin** | **Action** | **Output** |
| **1** |  | **Enter** | **Please Enter Pin on the numeric keypad.** |
| **2** | **1470** | **Enter** | **Incorrect Pin Try Again.** |
| **3** | **7382** | **Enter** | **Incorrect Pin Try Again.** |
| **4** | **8234** | **Enter** | **Program Exists if test case 2-4 executed consecutively.** |

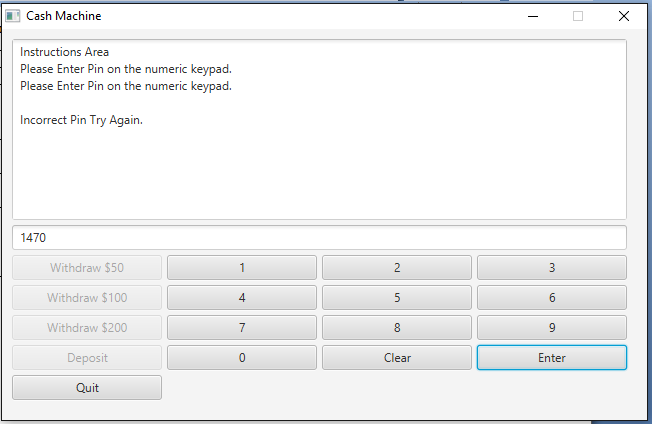
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATM Deposit Testing**  **Enter Pin 1234 and click Deposit and then input the following values.** | | | | |
| **Test Case #** | **Beginning Balance** | **Amount** | **Action** | **Output** |
| **1** | **$1,573.91** |  | **Enter** | **Please Enter Deposit Amount.** |
| **2** |  | **3000** | **Enter** | **Deposit Amount Must Be Less Than $2500.00** |
| **3** |  | **200** | **Enter** | **Deposit Successful**  **Deposit Amount: $200**  **New Balance: $1,773.91** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATM Withdraw Testing**  **Enter Pin 1234 and click the following buttons.** | | | | |
| **Test Case #** | **Beginning Balance** | **Amount** | **Action** | **Output** |
| **1** | **$1,573.91** | **50** | **Enter** | **Withdraw Successful**  **Withdraw Amount: $50.00**  **New Balance: $1,523.91** |
| **2** | **$1,523.91** | **100** | **Enter** | **Withdraw Successful**  **Withdraw Amount: $100.00**  **New Balance: $1,423.91** |
| **3** | **$1,423.91** | **200** | **Enter** | **Withdraw Successful**  **Withdraw Amount: $200.00**  **New Balance: $1,223.91** |

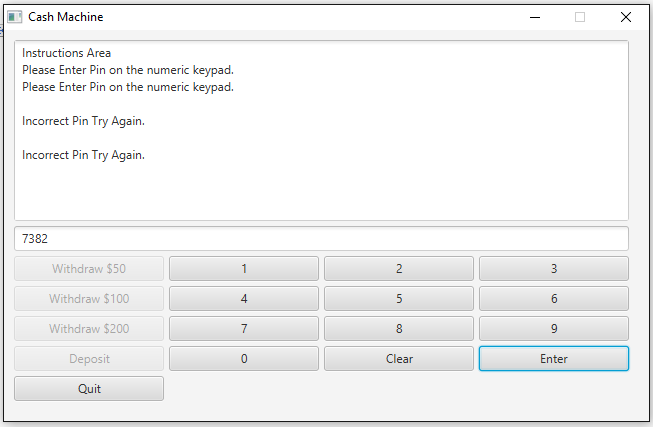
**6. Testing Output**

Pin Testing Test Case # 1

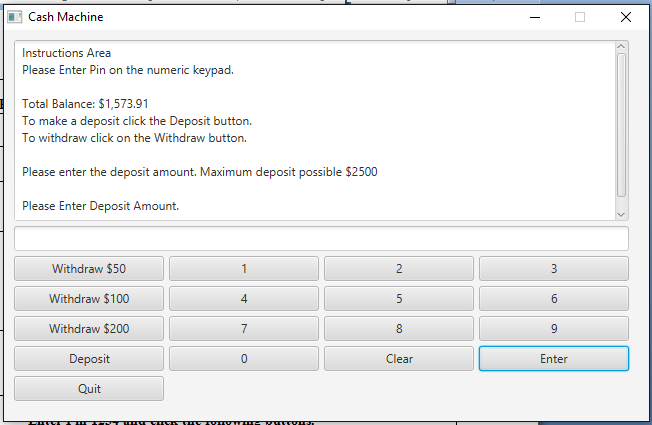


Pin Testing Test Case # 2

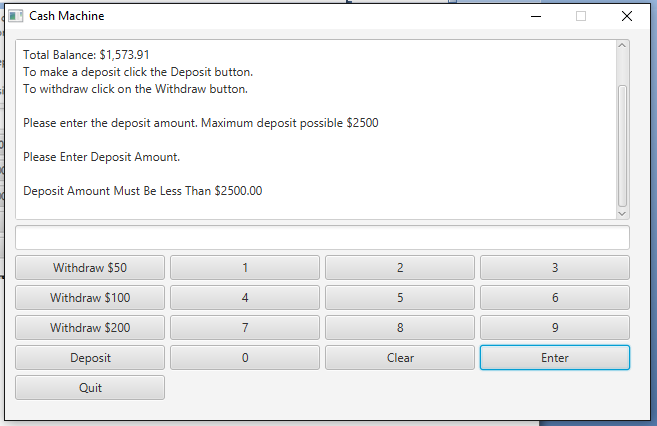
Pin Testing Test Case # 3



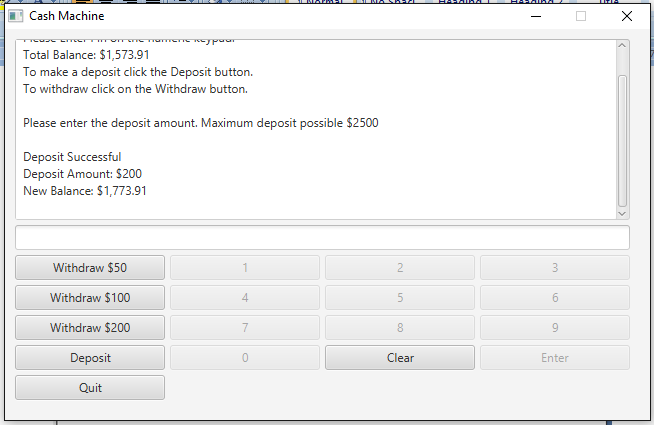
Deposit Testing Test Case # 1



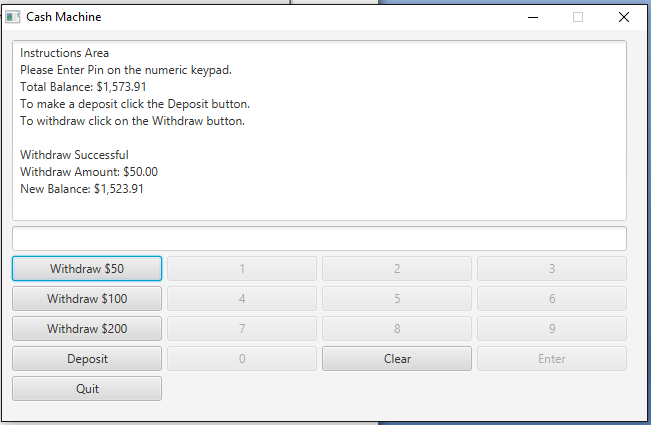
Deposit Testing Test Case # 2



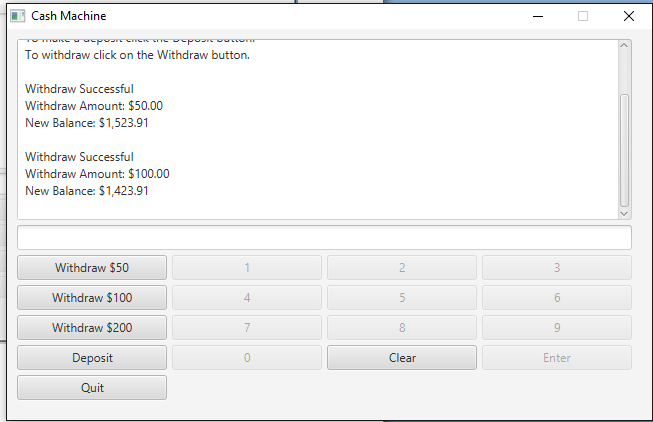
Deposit Testing Test Case # 3



Withdraw Testing Test Case # 1



Withdraw Testing Test Case # 2



Withdraw Testing Test Case # 3

