College

Boulevard de Maisonneuve O Montréal, QC H3A 1L2 – (514) 849.1234

AUTOMATIC TELLER SIMULATOR

Phase 1 Integration Project
PPIE V1-1

Student: Dangelo Trindade Santana
CDI college, programmer analyst /
Internet solutions developer - LEA.9c
dangelosantana@hotmail.com
https://ca.linkedin.com/in/dangelo-santana-21265a96

Professor: Lounis Zaidi
CDI COLLEGE, PROGRAM ANALYSIS SYSTEM
lounis.zaidi@collegecdi.ca

Montreal, January 2017

Content

1 Introduction

The present project aims to demonstrate the skills and knowledge acquired until the present moment in the programming course.

Simulating a real world situation the project highlight the first work day in the consulting firm, as junior programmer.

The main task received consists in a development of a new prototype application that simulates the operation of an automatic Teller Machine.

Applying a graphical interface and also allowing the practice of a lot of items already used and previous studied the main objective are listed below:

- Interpret specification and analysis performed;
- Design a solution based on the requirements and specification;
- Design the logic required for and event-driven solution;
- Translate design documents and algorithms into source code;
- Read from and write to a file;
- Use debugging tools, and error-handling techniques;
- Validate the solution with test data;
- Integrate the knowledge's acquired thus far;
- Have fun while programming with visual studio C#.net

2. Project Description

Based in the current version as "demo interface will be developed a new system and interface to deal with transactions of automatic Teller Machine.

2.1 Business Requirement Resume

General Description: "After successfully completing your studies, you have landed the job of your dreams with a consulting firm. Today is your first day on the job as a junior programmer. Your immediate supervisor gives you the responsibility to develop an application prototype that simulates the operation of an automatic teller machine (ATM). The client is running an older version and would like it updated. Your supervisor is expecting you to deliver a functioning prototype in six days. This is your opportunity to showcase your creative talents, and your analysis, design, coding, and testing abilities.

2.2 Function and Features:

2.2.1 Login: Before performing any transaction, the user must enter his or her name and on an input screen. Since the operation of this input screen should simulate the normal operation of an ATM, the PIN should not appear on the screen. In addition to the message which appears after every unsuccessful attempt, if after three tries the PIN matching the name has not been entered, the application should display a message requesting the user to try using the ATM again later. The names and PINs of users must be validated using data contained in the Customers.txt text file having the following structure:

- name (String)
- PIN (String – 4 characters)

2.2.2 Main Form: Once access has been authorized, the main form of the application should allow the user to carry out one of the following transactions:

- Deposit
- Withdrawal
- Transfer
- Bill payment

Below is the structure of the Accounts.txt sequential file in which account balances are stored:

- Account type (1 character)
- Pin (string – 4 characters)
- Account number (string – 5 characters)
- Account balance (~~single~~ Decimal)

2.2.3 Deposit: For a deposit, the user must enter the amount and, if required, be able to select the account type to be credited. The checking account is the default for this transaction.

2.2.4 Withdrawal: For a withdrawal, the user must enter the amount and, if required, be able to select the account type to be debited. The checking account is the default for this transaction subject to a maximum of $1,000. The ATM accepts only transactions for which the amount entered is a multiple of $10. There is no daily maximum amount (apart from the user's account balance).

2.2.5 Transfer: For a transfer, the user must enter the amount and the type of transfer (from checking to savings, or vice versa). This transaction is subject to a maximum $100,000. The system must allow only a transfer from checking to savings, or from savings to checking.

2.2.6 Bill Payment: For a bill payment, which is done from a checking account only, the user must enter the amount of the transaction. The checking account is debited by the same amount. In addition, a $1.25 fee is charged to the checking account. The maximum per transaction is $10,000.

3. General Rules

3.1 Chose Account: When a user performs an operation, the application should first ask if it should be done using a checking or savings account, and then ask for the transaction amount.

3.2 Check Account Balance: The application must check the account balance before doing a transaction. Any transaction that would result in a negative balance must be rejected.

3.3 Update Account Balance: The balance of the account affected by a transaction should be updated and displayed after each transaction.

3.4 Multi transaction: The user should be able to do as many transactions as he or she would like to do before leaving the ATM.

3.4 No Money: A warning message should inform the user that the ATM can no longer carry out withdrawals when there is no money available. When a withdrawal transaction event occurs for an amount greater than the balance remaining in the ATM, the ATM should advise the user they can charge the transaction amount to the amount still available in the user's account.

3.5 Daily Refilled: When the application is initiated at start of each day, the bank's balance money is automatically refilled with up to $5,000 for a maximum of $20,000.

4. System Administrator

The system administrator, as any other user, must enter his or her name and PIN (personal identification number) on the same input screen. The system administrator may perform only system transactions (he or she has no personal account).

The supervisor can cause interest to be paid to all savings accounts at the rate of 1% (Balance * rate/365/100).

Once access has been authorized, a special menu is displayed. This menu offers the following options:

- Pay interest
- Refill the ATM with money
- Take the ATM out of service
- Print the accounts report

The system administrator puts in more money in batches of $5,000. There should not be more than $20,000 available in the ATM.

5. Modeling ATM requirements:

5.1.1 Data Dictionary Class Customer:

Data Element Name	Customer Name	Data Element Name	Customer PIN
Alias	name	Alias	pinNumber
Type	string	Type	string
Length	20	Length	4
Range of values		Range of values	
Definition		Definition	

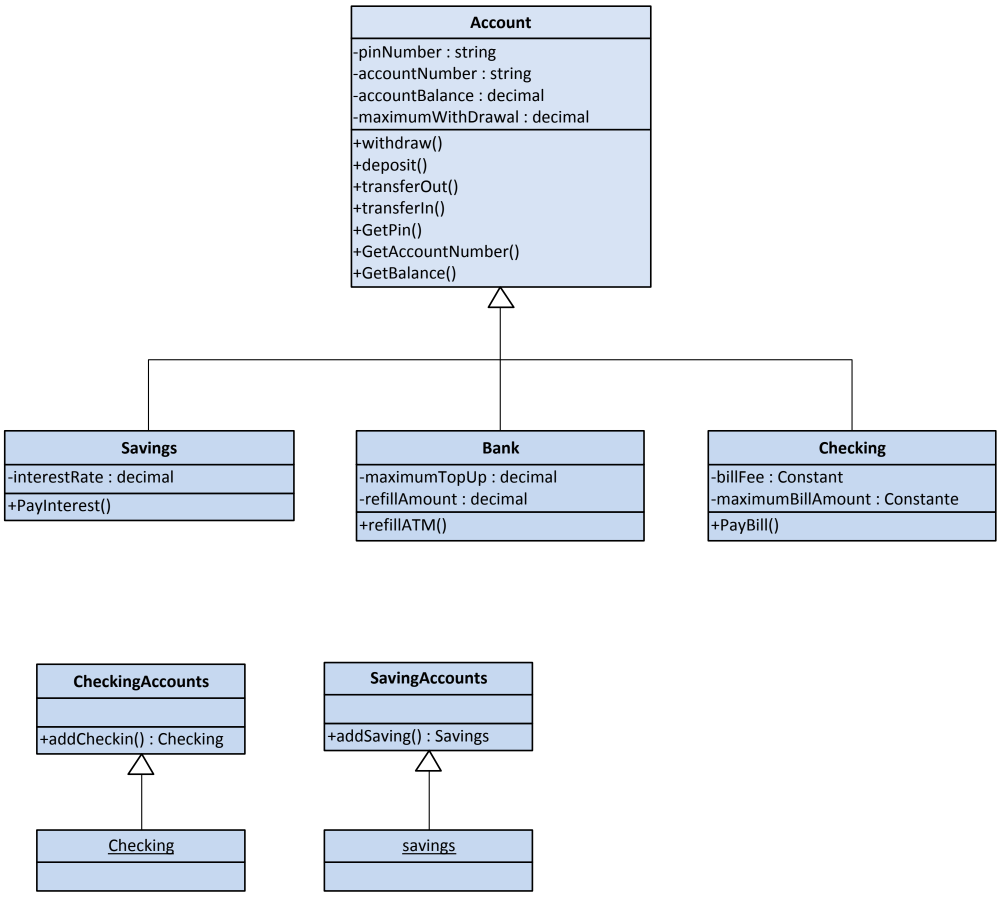
5.1.2 Data Dictionary Class Account:

Data Element Name [key]	Personal Identification Number	Data Element Name	Account Number
Alias	pinnumber	Alias	acNumber
Type	string	Type	string
Length	4	Length	5
Range of values		Range of values	
Definition		Definition	

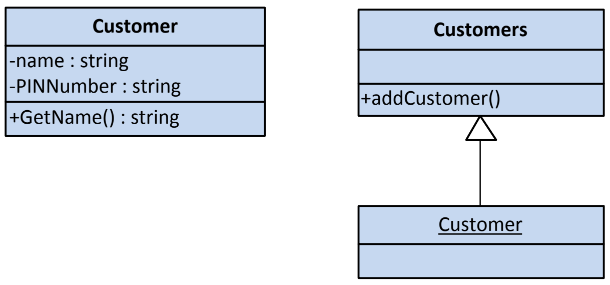
Data Element Name	Account Balance	Data Element Name	Account Type
Alias	acBalance	Alias	acType
Type	Decimal	Type	char
Length		Length	
Range of values		Range of values	“B”, “S”, “C”
Definition		Definition	

5.2 Class Diagram:

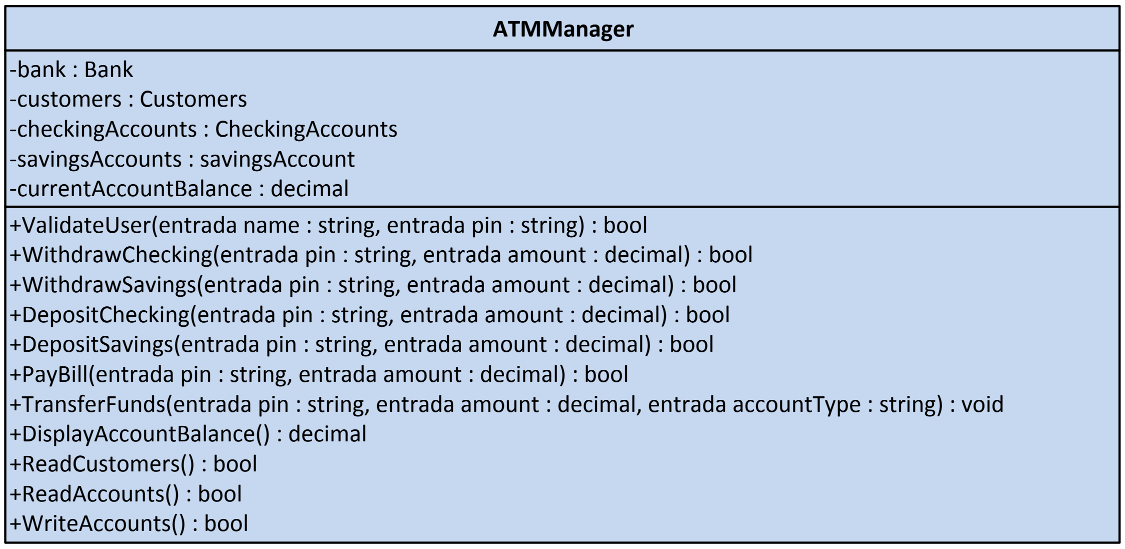
5.2.1 Accounts Class Diagram:



5.2.2 Customer Class Diagram:



5.2.3 “ATMManager” Class Diagram:



ATMManager

-bank : Bank
-customers : Customers
-checkingAccounts : CheckingAccounts
-savingsAccounts : savingsAccount
-currentAccountBalance : decimal

+ValidateUser(entrada name : string, entrada pin : string) : bool
+WithdrawChecking(entrada pin : string, entrada amount : decimal) : bool
+WithdrawSavings(entrada pin : string, entrada amount : decimal) : bool
+DepositChecking(entrada pin : string, entrada amount : decimal) : bool
+DepositSavings(entrada pin : string, entrada amount : decimal) : bool
+PayBill(entrada pin : string, entrada amount : decimal) : bool
+TransferFunds(entrada pin : string, entrada amount : decimal, entrada accountType : string) : void
+DisplayAccountBalance() : decimal
+ReadCustomers() : bool
+ReadAccounts() : bool
+WriteAccounts() : bool

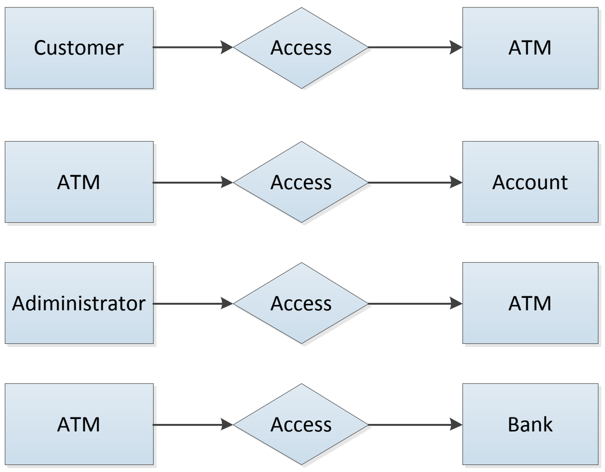
5.2.4 Some hints:

ReadAccounts() returns Boolean - do not read the first character (B, C or S) found in the Accounts.txt into the object instances. Use it to determine to which collection the account should be added (Bank, checkingAccounts, savingAccounts)

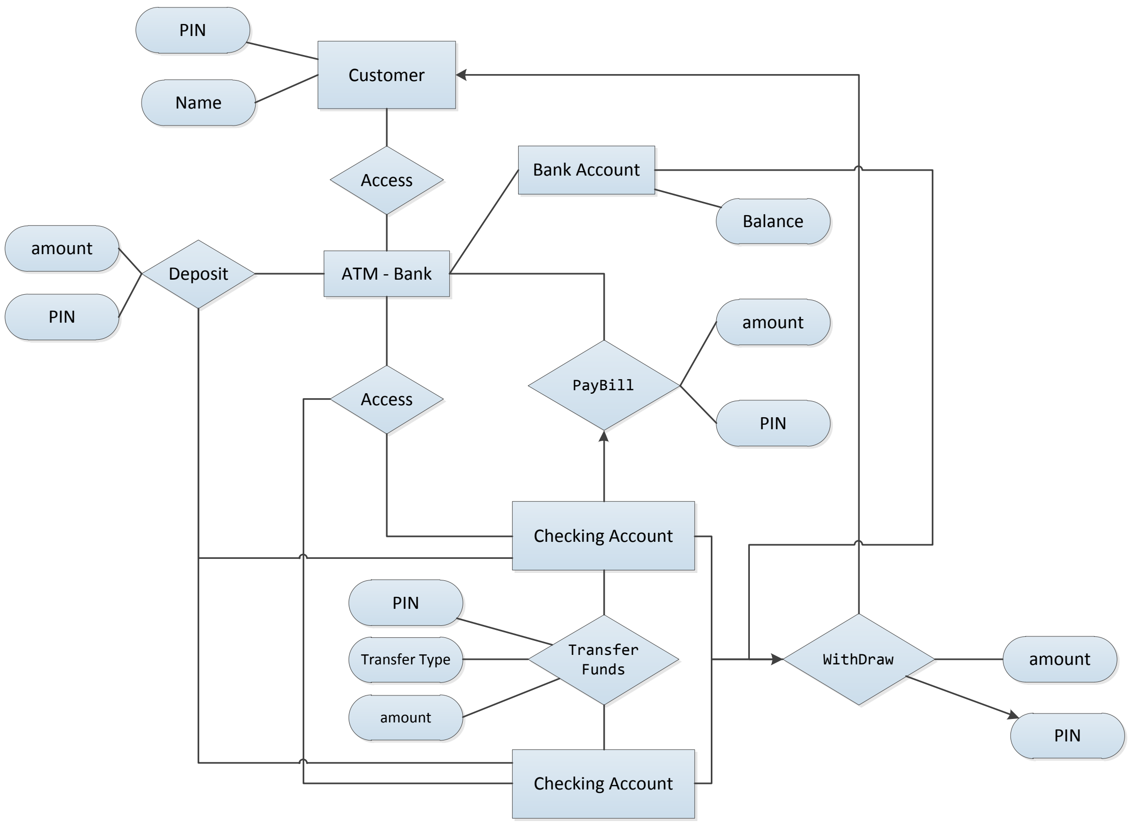
WriteAccounts() returns Boolean - write the bank, all checkingAccounts, and all savingsAccounts to Accounts.txt. - add a first character (B, C or S) to the beginning of each account string before writing to Accounts.txt.

5.3 Relationship Entity Diagram:
Once that was complete detailed relationship, classes diagram and entities dictionary. The cardinality was not detailed in ERD diagrams below.

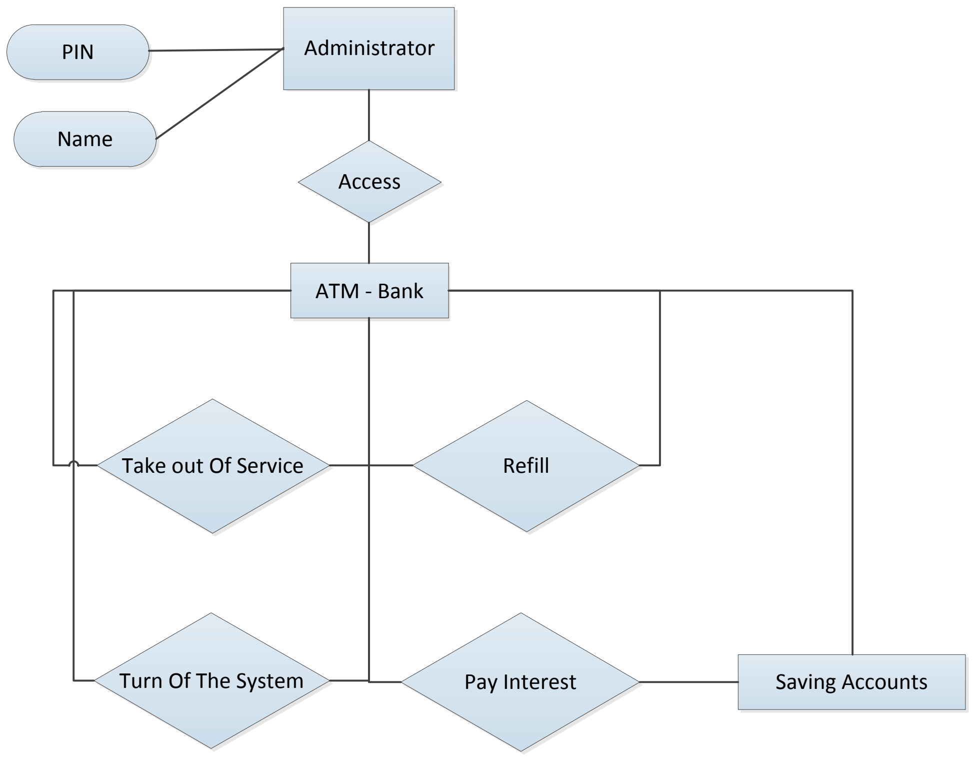
5.3.1 Individual ERD



5.3.2 Customer Segment ERD

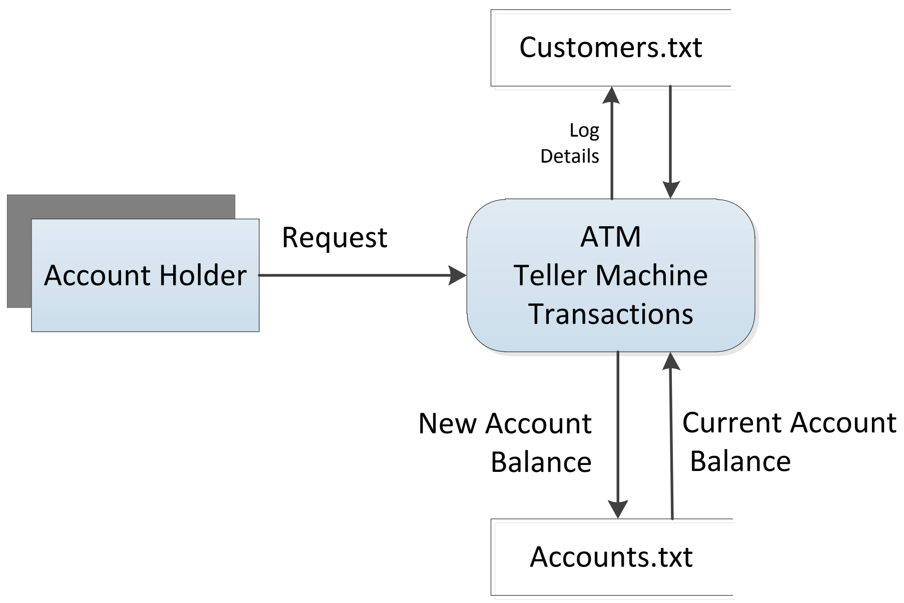


5.3.2 Administrator Segment ERD

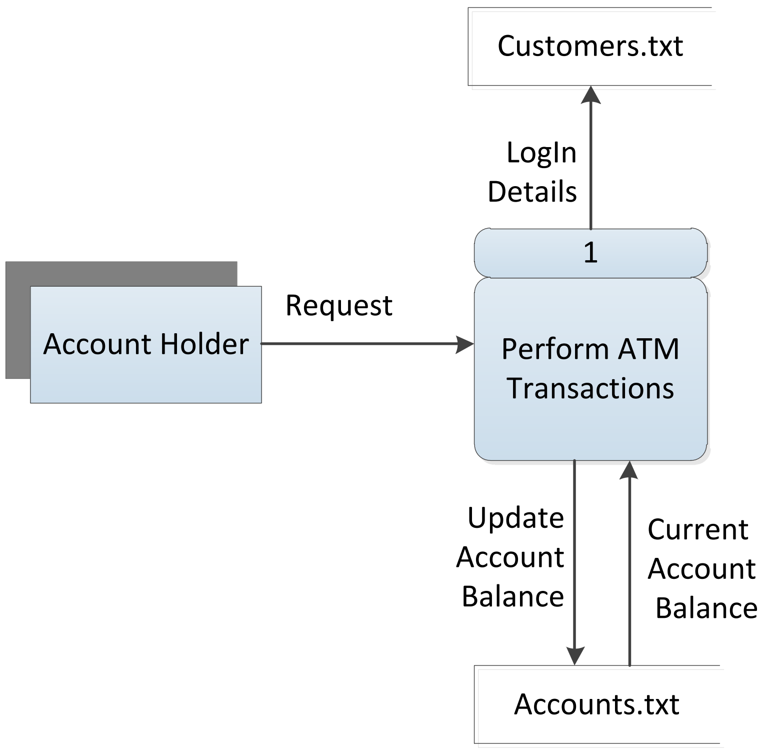


5.4 Data Flow Diagram:

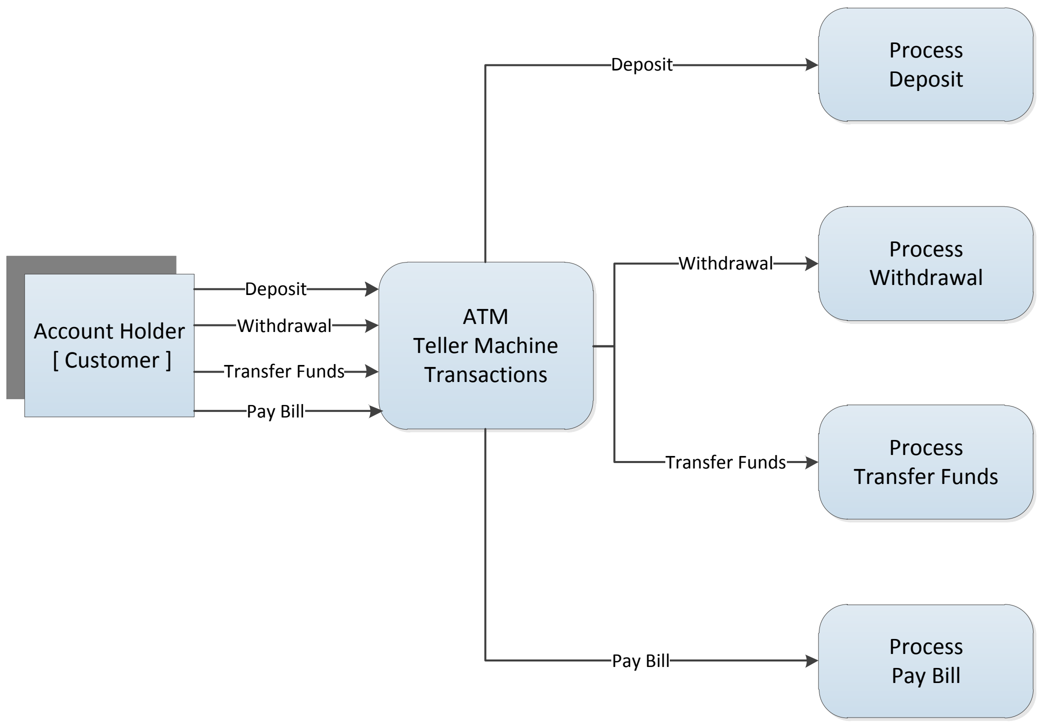
5.4.1 Data Flow Diagram Level-Zero



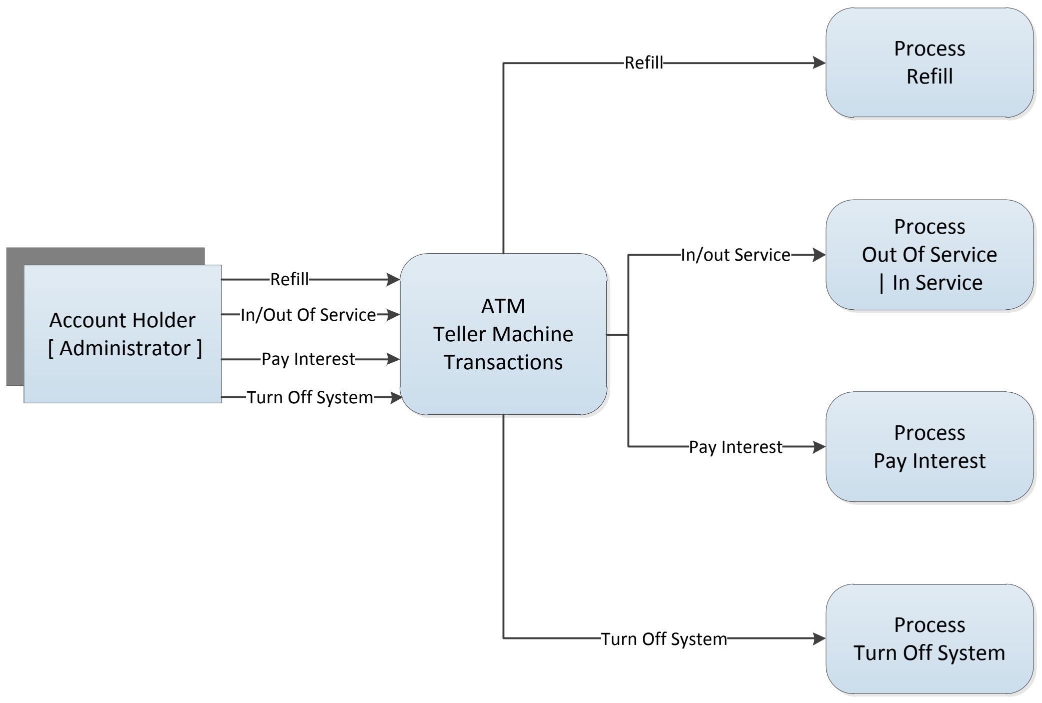
5.4.2 Data Flow Diagram Level-one



5.4.3 Customer Data Flow Diagram DFD Simplified Transactions Flow



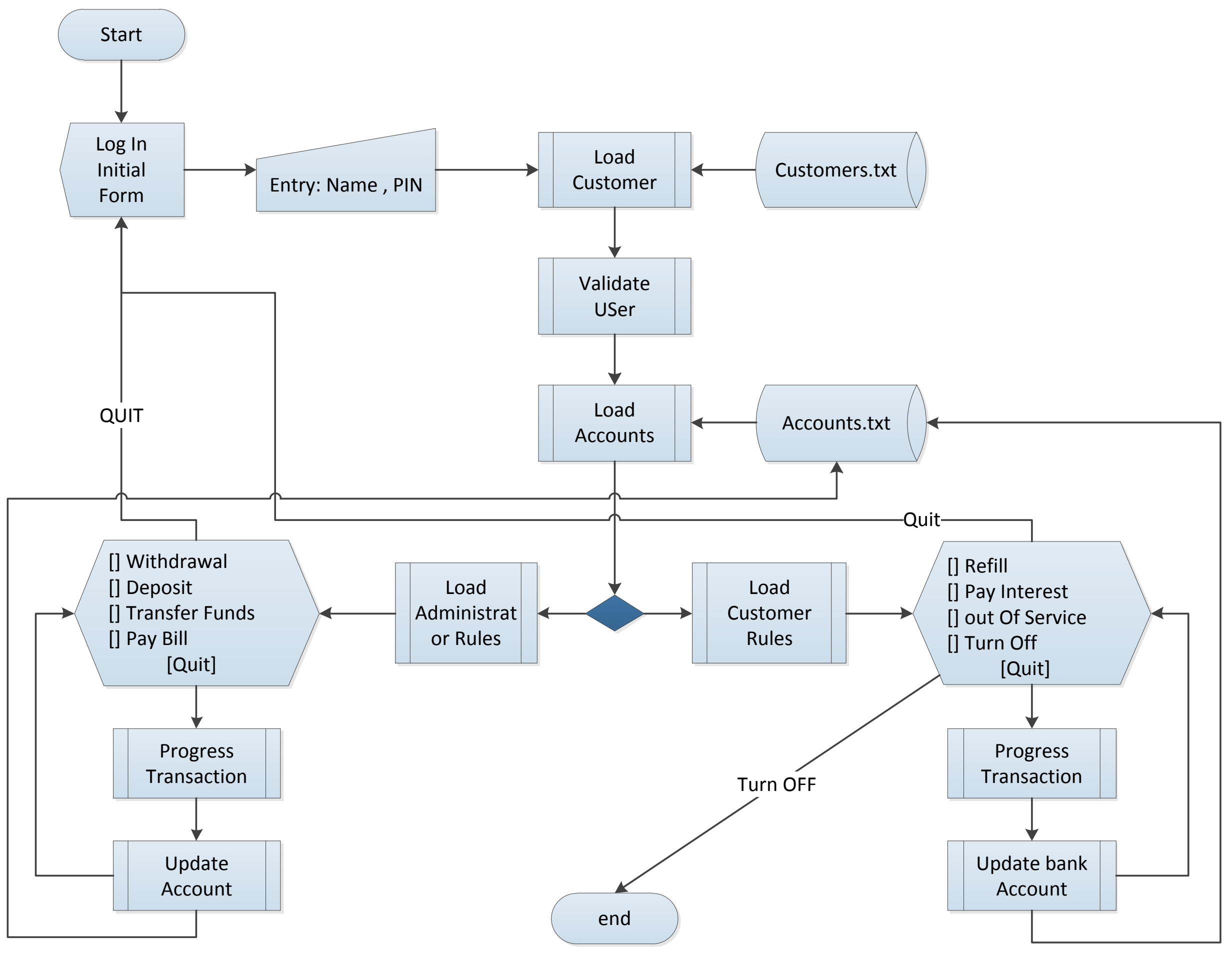
5.4.4 Customer Data Flow Diagram DFD Simplified Transactions Flow



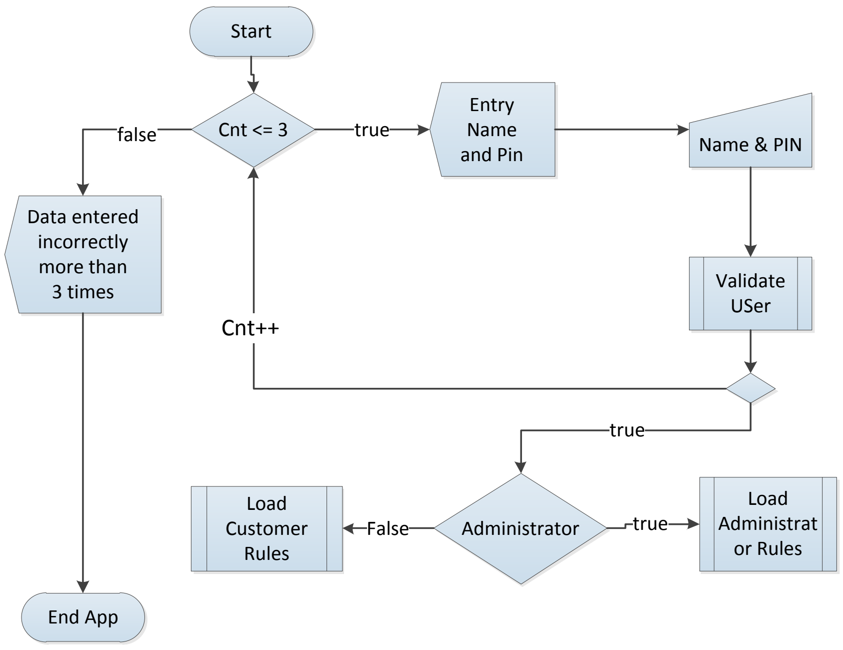
6. Program and Logic Design:

6.1 Flow Chart and pseudo code:

6.1.1 Hierarchy chart Process and Sub Process



6.1.2 Log In Flow Chart



6.1.3 Withdrawal Pseudo Code

```
             Withdrawal (Account Method)
VARIABLES
    Amount IS Decimal, PIN integer
BEGIN
    Input Amount from customer keypad
    read customer's details from Accounts.txt database
    IF customer has insufficient funds
        THEN IF customer has zero funds
          THEN
                Display 'insufficient funds' message
        END
    ELSE IF amount < 10,000.00
      THEN Display 'Withdrawal Limit' message
    END
    ELSE IF amount % 10 != 0
      THEN Display 'ten multiples' message
    END
    ELSE BEGIN
        Display offer of available funds
        THEN set reduced Amount from customer Account
      THEN set reduced Amount from ATM – Bank Account
        THEN dispense cash Amount
        THEN update accounts database
        END
    ENDIF
END Withdrawal
```

6.1.4 Deposit Pseudo Code

```
             Deposit (Account Method)
VARIABLES
Decimal Amount, integer PIN
BEGIN Deposit
IINPUT amount, pin
READ account details from Accounts.txt
IF radio button "Account Checking" Checked
      IF PIN == CheckingAccount.PinNumber
      THEN
      Account balance += amount
      Display deposit authorized
ELSE IF amount || Pin == Null
      THEN
      Display error validation message
      END IF
ELSE  IF PIN == SavingAccount.PinNumber
      THEN
      Account balance += amount
      Display deposit authorized
END IF
END DEPOSIT
```

6.2.5 Refill ATM Machine Pseudo Code

```
             Refill ATM Machine
VARIABLES
Decimal Bundle = 5000, Decimal MaximumTopUp = 20000,
Decimal AcountBalance
BEGIN Refill
READ account details from Accounts.txt
WHILE (AcountBalance + Bundle <= MaximumTopUp)
      THEN AcountBalance += Bundle
END WHILE
END Refill
```

CDI College

7. Source Code:

7.1 Login Form Source Code:

Login Form Source Code

```
using System.Windows.Forms;
using ATMclassLibrary;
using System.Threading;

namespace ATMClienteApp
{
public partial class FrmLogin : Form
{
int cnt = 1;
public bool outofservice = false;
public FrmLogin()
{
InitializeComponent();
}
private void FrmLogin_Load(object sender, EventArgs e)
{
OutofServiceMode();
}
private void OutofServiceMode()
{
if (outofservice==false)
{
lboutofService.Visible = false;
lbsorry.Visible = false;
}
else
{
lboutofService.Visible = true;
lbsorry.Visible = true;
}}
public void btLogin_Click(object sender, EventArgs e)
{
atmManager atm = new atmManager();
atm.loadBank();
atm.loadCustomer();
string name = txbName.Text;
string pin = txbPin.Text;
bool loginconfirm = atm.validateUser(name, pin);
if (outofservice == false || name == "Korben Dallas")
{
if (txbName.Text == "" || txbPin.Text == "")
{
MessageBox.Show("Please entry [Name] and [PApssword]", "Error: Missing Data",
MessageBoxButtons.OK, MessageBoxIcon.Error);
this.txbPin.Text = "";
this.txbName.Text = "";
txbPin.Clear();
txbName.Clear();
}
else if (loginconfirm == true && name != "Korben Dallas")
{
```

```
this.txbPin.Text = "";
this.txbName.Text = "";
FrmAtmMain frmMain = new FrmAtmMain(pin, this, atm);
this.Hide();
frmMain.Show();
frmMain.Activate();
}
else if (loginconfirm == true && name == "Korben Dallas")
{
this.txbPin.Text = "";
this.txbName.Text = "";
FrmAdmin frmAdmin = new FrmAdmin(pin, this, atm);
this.Hide();
frmAdmin.Show();
frmAdmin.Activate();
}
else if (loginconfirm == false && cnt < 3)
{
cnt++;
MessageBox.Show("Sorry! \n The User or Password is incorrect", "Error: User || Passwor",
MessageBoxButtons.OK, MessageBoxIcon.Error);
txbPin.Clear();
}
else if (loginconfirm == false && cnt >= 3)
{
MessageBox.Show("Sorry! \n Was not possible access your account \n try again later, Thank
you", "Error: User || Passwor", MessageBoxButtons.OK, MessageBoxIcon.Error);
this.txbPin.Text = "";
this.txbName.Text = "";
txbPin.Clear();
txbName.Clear();
}}
else
{
txbName.Clear();
txbPin.Clear();
for (int i = 1; i < 5; i++)
{
Thread.Sleep(1000);
Console.Beep();
lboutofService.Show();
lbsorry.Show();
}
lboutofService.Show();
lbsorry.Show();
}}
private void btLogin_KeyDown(object sender, KeyEventArgs e)
{
if (e.KeyCode == Keys.Enter)
{
btLogin.PerformClick();
e.SuppressKeyPress = true;
e.Handled = true;
}}
private void txbPin_TextChanged(object sender, EventArgs e){}
private void txbPin_KeyDown(object sender, KeyEventArgs e)
{
if (e.KeyCode == Keys.Enter)
{
btLogin.PerformClick();
```

```
e.SuppressKeyPress = true;
e.Handled = true;
}}
private void txbtime_TextChanged(object sender, EventArgs e){}
private void lboutofService_Click(object sender, EventArgs e)
{}}}
```

7.2 Customer Form Source Code:

Customer Form Source Code

```
…
using System.Threading.Tasks;
using System.Windows.Forms;
using ATMclassLibrary;

namespace ATMClienteApp
{
public partial class FrmAtmMain : Form
{
string pin;
FrmLogin mylog;
atmManager atm = new atmManager();
decimal availabelATMmoney;
//string currText;
//string oldText;
public FrmAtmMain(string PIN, FrmLogin MYLOG, atmManager ATM)
{
InitializeComponent();
pin = PIN;
mylog = MYLOG;
atm = ATM;
}
private void btPayBill_Click(object sender, EventArgs e)
{
if (txbAmount.Text != null && txbAmount.Text != "")
{
decimal billvalue = Convert.ToDecimal(txbAmount.Text);
if (billvalue > 10000)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = "The Pay Bill Transaction do not progress values upper to
$10,000." + Environment.NewLine + Environment.NewLine + "Would Like to progress a
new Transaction?";
}
else if (txbAmount.Text == null)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}
else if (rdbtFromChecking.Checked == true)
{
if (atm.checkingAccount.acBalance - 1.25m - billvalue < 0)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Sorry, There is not available money to
progress this transaction." + Environment.NewLine;
}
else
{
atm.checkingAccount.paybill(billvalue);
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Your payment Was successfully
progressed." + Environment.NewLine + Environment.NewLine + "Thank you for use the
ATM Teller MAchine.";
writeupdate();}}
else if (rdbtFromChecking.Checked != true)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = "The Pay Bill Transaction can only be progress from Checking
Account." + Environment.NewLine + "please Select Checking Account, and try again";
}
riseNewtransaction();
}
else
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}}
private void btCloseApp_Click(object sender, EventArgs e)
{
txbAccountinfo.Clear();
txbAmount.Clear();
atm.bank.acBalance = availabelATMmoney;
writeupdate();
this.Close();
mylog.Show();
}
public void FrmAtmMain_Load(object sender, EventArgs e)
{
DisplayAccount();
rdbtFromChecking.Checked = true;
btNewTransaction.Hide();
availabelATMmoney = atm.bank.acBalance;
```

```
txbAmount.Focus();
}
private void btDeposit_Click(object sender, EventArgs e)
{
if (txbAmount.Text != null && txbAmount.Text != "")
{
decimal amount = Convert.ToDecimal(txbAmount.Text);
if (rdbtFromChecking.Checked == true)
{
atm.depositChecking(pin, amount);
txbAccountinfo.Text = "Deposit Autorized, Value: " + amount.ToString() +
Environment.NewLine + " to checking Account:" +
atm.checkingAccount.acNumber.ToString() + Environment.NewLine +
Environment.NewLine + " Please Insert The Envelope!";
riseNewtransaction();
writeupdate();
}

else
{
atm.depositsaving(pin, amount);
txbAccountinfo.Text = "Deposit Autorized, Value: " + amount.ToString() +
Environment.NewLine + " to Saving Account:" +
atm.savingAccount.acNumber.ToString() + Environment.NewLine +
Environment.NewLine + " Please Insert The Envelope!";
riseNewtransaction();
writeupdate();
}}
else
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}}
public void DisplayAccount()
{
txbAccountinfo.Clear();
decimal chebal = atm.checkingAccount.acBalance;
decimal savbal = atm.savingAccount.acBalance;
decimal totbal = chebal + savbal;
txbAccountinfo.Text = " Welcome " + atm.customer.Name + Environment.NewLine +
Environment.NewLine + " Checking Account [" +
atm.checkingAccount.acNumber.ToString() + "]" + Environment.NewLine + " Current
Balance: " + chebal.ToString() + Environment.NewLine + Environment.NewLine + "
Saving Account [" + atm.savingAccount.acNumber.ToString() + "]" +
Environment.NewLine + " Current Balance: " + savbal.ToString() +
Environment.NewLine + Environment.NewLine + " Total amount disponible: " +
totbal.ToString() + Environment.NewLine + Environment.NewLine + "In Order To
Tranfer Funds Select: " + Environment.NewLine + "[] Checking → Checking to Saving
Account;" + Environment.NewLine + "[] Saving → Saving to Checking Account; ";
txbAmount.Focus();
}
private void btWithdrawal_Click(object sender, EventArgs e)
{
decimal amount = Convert.ToDecimal(txbAmount.Text);
if (amount % 10 != 0 || amount > 1000)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Sorry, the amount entered must be a
multiple of $10, Limited to $1000.00 for each transaction";
riseNewtransaction();
}
else if (availabelATMmoney < amount)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Sorry, there is not enough money
available to progress this transaction ";
riseNewtransaction();
}
else if (txbAmount.Text == null)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}
else if (rdbtFromChecking.Checked == true)
{
atm.ChekingWithdraw(pin, amount);
availabelATMmoney -= amount;
txbAccountinfo.Text = "Was withdrawn the amount of $" + amount +
Environment.NewLine + "the Currfent Balance is: " +
atm.checkingAccount.acBalance.ToString();
riseNewtransaction();
}
else if (rdbtFromChecking.Checked == true)
{
atm.savingWithdraw(pin, amount);
availabelATMmoney -= amount;
txbAccountinfo.Text = "Was withdrawn the amount of $" + amount +
Environment.NewLine + "the Current Balance is: " + atm.savingAccount.ToString();
riseNewtransaction();}}
public void riseNewtransaction()
{
btNewTransaction.Show();
```

```
btDeposit.Hide();
btPayBill.Hide();
btWithdrawal.Hide();
btTranfFunds.Hide();
bgaccountSet.Enabled = false;
bgAmount.Enabled = false;
btNewTransaction.Focus();
}
private void btTranfFunds_Click(object sender, EventArgs e)
{
if (txbAmount.Text != null && txbAmount.Text != "")
{
txbAccountinfo.Clear();
decimal checbal = atm.checkingAccount.acBalance;
decimal savbal = atm.savingAccount.acBalance;
decimal amount = Convert.ToDecimal(txbAmount.Text);
if (amount > 100000)
{
txbAccountinfo.Text = Environment.NewLine + "Sorry, the amount transaction is Limited
to $100,000.00";
riseNewtransaction();
}
else if (txbAmount.Text == null)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}
else if (rdbtFromChecking.Checked == true)
{
if (checbal - amount < 0)
{
txbAccountinfo.Text = Environment.NewLine + "Sorry, there is not enough balance to
progress this transaction";
riseNewtransaction();
}
else
{
atm.transferFunds(pin, amount, "CS");
txbAccountinfo.Text = Environment.NewLine + "Transfer transaction successfully
progressed" + Environment.NewLine + Environment.NewLine + "New Checking
Balance: " + atm.checkingAccount.acBalance.ToString() + Environment.NewLine +
Environment.NewLine + "New Saving Balance: " +
atm.savingAccount.acBalance.ToString();
riseNewtransaction();
writeupdate();
}}
else if (rdbtFromSaving.Checked == true)
{
if (savbal - amount < 0)
{
txbAccountinfo.Text = Environment.NewLine + "Sorry, there is not enough balance to
progress this transaction";
riseNewtransaction();
writeupdate();
}
else
{
atm.transferFunds(pin, amount, "SC");
riseNewtransaction();
txbAccountinfo.Text = Environment.NewLine + "Transfer transaction successfully
progressed" + Environment.NewLine + Environment.NewLine + "New Checking
Balance: " + atm.checkingAccount.acBalance.ToString() + Environment.NewLine +
Environment.NewLine + "New Saving Balance: " +
atm.savingAccount.acBalance.ToString();
riseNewtransaction();
writeupdate(); }}}
else
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}}
private void btNewTransaction_Click_1(object sender, EventArgs e)
{
writeupdate();
txbAmount.Clear();
DisplayAccount();
btNewTransaction.Hide();
btDeposit.Show();
btPayBill.Show();
btWithdrawal.Show();
btTranfFunds.Show();
bgaccountSet.Enabled = true;
bgAmount.Enabled = true;
txbAmount.Focus();
}
private void txbAmount_KeyPress(object sender, KeyPressEventArgs e)
{
if (!char.IsDigit(e.KeyChar))
{
if ((char)e.KeyChar != '.' && e.KeyChar != 8)
e.Handled = true;
}
```

```
if (e.KeyChar == '.' && this.txbAmount.Text.IndexOf('.') >= 0)
e.Handled = true;
int index = this.txbAmount.Text.IndexOf('.');
if (index >= 0)
{
string Size = this.txbAmount.Text.Substring(index);
if (Size.Length > 2 && e.KeyChar != 8)
{
e.Handled = true;
}}}
private void txbAmount_TextChanged(object sender, EventArgs e){ }
private ClassMasc deploymasc(char charac, string text)
{
// not deployed or used yet
ClassMasc mascobject = new ClassMasc();
mascobject.receiveKey(charac);
mascobject.ReceiveWord(text);
return mascobject;
}
private void txbAmount_Click(object sender, EventArgs e)
{
txbAmount.SelectionStart = txbAmount.Text.Length + 1;
}
private void btWithdrawal_Click_1(object sender, EventArgs e)
{
if (txbAmount.Text != null && txbAmount.Text != "")
{
decimal amount = Convert.ToDecimal(txbAmount.Text);
if (amount % 10 != 0 || amount > 1000)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Sorry, the amount entered must be a
multiple of $10, Limited to $1000.00 for each transaction";
riseNewtransaction();
}
else if (availabelATMmoney < amount)
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Sorry, there is not enough money
available to progress this transaction ";
riseNewtransaction();
}
else if (rdbtFromChecking.Checked == true)
{
decimal WithDrawSimulation = atm.checkingAccount.acBalance - amount;
if (WithDrawSimulation >= 0)
{
atm.ChekingWithdraw(pin, amount);
availabelATMmoney -= amount;
txbAccountinfo.Text = "Was withdrawn the amount of $" + amount +
Environment.NewLine + "the Currfent Balance is: " +
atm.checkingAccount.acBalance.ToString();
riseNewtransaction();
atm.bank.acBalance = availabelATMmoney;
writeupdate();
}
else
{
txbAccountinfo.Text = " There is not availabel money at your account to progress this
operation" + amount + Environment.NewLine + "the Currfent Balance is: " +
atm.checkingAccount.acBalance.ToString();
riseNewtransaction();}}
else if (rdbtFromSaving.Checked == true)
{
decimal WithDrawSimulation = atm.checkingAccount.acBalance - amount;
if (WithDrawSimulation >= 0)
{
atm.savingWithdraw(pin, amount);
availabelATMmoney -= amount;
txbAccountinfo.Text = "Was withdrawn the amount of $" + amount +
Environment.NewLine + "the Current Balance is: " +
atm.savingAccount.acBalance.ToString();
riseNewtransaction();
atm.bank.acBalance = availabelATMmoney;
writeupdate();
}
else
{
txbAccountinfo.Text = " There is not availabel money at your account to progress this
operation" + amount + Environment.NewLine + "the Currfent Balance is: " +
atm.checkingAccount.acBalance.ToString();
riseNewtransaction();}}}
else
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction();
}}
public void writeupdate()
{
atm.WriteAccount();
}
private void bttestwrite_Click(object sender, EventArgs e)
{
writeupdate();
```

```
}
private void txbAmount_ImeModeChanged(object sender, EventArgs e)
{}
public void validateAmount()
{
if (txbAmount.Text == null || txbAmount.Text == "")
{
txbAccountinfo.Clear();
txbAccountinfo.Text = Environment.NewLine + "Please enter the amount in order to
progress this transaction";
riseNewtransaction(); }}}}
```

7.3 Administrator Form Source Code:

Administrator Form Source Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ATMclassLibrary;
using System.Threading;

namespace ATMClienteApp
{
public partial class FrmAdmin : Form
{
string pin;
FrmLogin mylog;
atmManager atm = new atmManager();
decimal availabelATMmoney;
string nnline = Environment.NewLine + Environment.NewLine;
string today = DateTime.Now.ToShortDateString();
public FrmAdmin(string PIN, FrmLogin MYLOG, atmManager ATM)
{
InitializeComponent();
pin = PIN;
mylog = MYLOG;
atm = ATM;
}
private void DisplayAtm()
{
txbadminInfo.Clear();
txbadminInfo.Text = Environment.NewLine + "welcome To ATM Administrator System" + nnline
+ "The current Balance at " + today + " Is: $" + atm.bank.acBalance.ToString();
}
private void FrmAdmin_Load(object sender, EventArgs e)
{
availabelATMmoney = atm.bank.acBalance;
DisplayAtm();
//btPrintRport.Enabled = false;
btNewOp.Hide();
```

```
}
private void btQuit_Click(object sender, EventArgs e)
{
this.Close();
mylog.Show();
WriteUpdateAcount();
}
private void btPayInterest_Click(object sender, EventArgs e)
{
foreach (Saving sv in atm.savingAccounts)
{
sv.payinterest();
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Interest Payment successfully progressed!" + nnline + "
Thank you for Use ATM TellerMachine Simulator";
risenewoperation();
//atm.bank.acBalance -= sv.payinterest();
WriteUpdateAcount();
}}
public void risenewoperation()
{
btPayInterest.Hide();
btDisplayReport.Hide();
btPrintRport.Hide();
btRefill.Hide();
btTakeOutofService.Hide();
btTurnOff.Hide();
btNewOp.Show();
}
private void btNewOp_Click(object sender, EventArgs e)
{
DisplayAtm();
btPayInterest.Show();
btDisplayReport.Show();
btPrintRport.Show();
btRefill.Show();
btTakeOutofService.Show();
btTurnOff.Show();
btNewOp.Hide();
}

public void WriteUpdateAcount()
{
atm.WriteAccount();
}
private void btRefill_Click(object sender, EventArgs e)
{
atm.bank.refillATM();
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Refill successfully progressed!" + nnline + " The current
available value on the Teller Machine Is: $" + atm.bank.acBalance + nnline + " Thank you
for Use ATM TellerMachine Simulator";
risenewoperation();
WriteUpdateAcount();
}
private void btTurnOff_Click(object sender, EventArgs e)
{
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " The System is Running the Safety turn off, do not power
off or unplug your machine!" + nnline + nnline + " Thank you for Use ATM TellerMachine
```

```
Simulator";

timer1.Enabled = true;
timer1.Interval = 5000;
timer1.Start();
timer1.Tick += new EventHandler(timer1_Tick);
}
private void timer1_Tick(object sender, EventArgs e)
{
txbadminInfo.Clear();
Application.Exit();
}
private void btPrintRport_Click(object sender, EventArgs e)
{
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Sorry this Function Is Temporarily out of Service";
risenewoperation();
}
private void btDisplayReport_Click(object sender, EventArgs e)
{
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Sorry this Function Is Temporarily out of Service";
risenewoperation();
}
private void btTakeOutofService_Click(object sender, EventArgs e)
{
FRMOutOfService outofservice = new FRMOutOfService(atm,mylog);
outofservice.Show();
outofservice.Show();
}}}
```

7.4 Out Of Service Form Source Code:

Out Of Service Form Source Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ATMclassLibrary;
using System.Threading;

namespace ATMClienteApp
{
public partial class FrmAdmin : Form
{
string pin;
FrmLogin mylog;
atmManager atm = new atmManager();
decimal availabelATMmoney;
string nnline = Environment.NewLine + Environment.NewLine;
string today = DateTime.Now.ToShortDateString();
public FrmAdmin(string PIN, FrmLogin MYLOG, atmManager ATM)
```

```
{
InitializeComponent();
pin = PIN;
mylog = MYLOG;
atm = ATM;
}
private void DisplayAtm()
{
txbadminInfo.Clear();
txbadminInfo.Text = Environment.NewLine + "welcome To ATM Administrator System" + nnline
+ "The current Balance at " + today + " Is: $" + atm.bank.acBalance.ToString();
}
private void FrmAdmin_Load(object sender, EventArgs e)
{
availabelATMmoney = atm.bank.acBalance;
DisplayAtm();
//btPrintRport.Enabled = false;
btNewOp.Hide();
}
private void btQuit_Click(object sender, EventArgs e)
{
this.Close();
mylog.Show();
WriteUpdateAcount();
}
private void btPayInterest_Click(object sender, EventArgs e)
{
foreach (Saving sv in atm.savingAccounts)
{
sv.payinterest();
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Interest Payment successfully progressed!" + nnline + "
Thank you for Use ATM TellerMachine Simulator";
risenewoperation();
//atm.bank.acBalance -= sv.payinterest();
WriteUpdateAcount();
}}
public void risenewoperation()
{
btPayInterest.Hide();
btDisplayReport.Hide();
btPrintRport.Hide();
btRefill.Hide();
btTakeOutofService.Hide();
btTurnOff.Hide();
btNewOp.Show();
}
private void btNewOp_Click(object sender, EventArgs e)
{
DisplayAtm();
btPayInterest.Show();
btDisplayReport.Show();
btPrintRport.Show();
btRefill.Show();
btTakeOutofService.Show();
btTurnOff.Show();
btNewOp.Hide();
}
public void WriteUpdateAcount()
{
```

```
atm.WriteAccount();
}
private void btRefill_Click(object sender, EventArgs e)
{
atm.bank.refillATM();
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Refill successfully progressed!" + nnline + " The current
available value on the Teller Machine Is: $" + atm.bank.acBalance + nnline + " Thank you
for Use ATM TellerMachine Simulator";
risenewoperation();
WriteUpdateAcount();
}
private void btTurnOff_Click(object sender, EventArgs e)
{
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " The System is Running the Safety turn off, do not power
off or unplug your machine!" + nnline + nnline + " Thank you for Use ATM TellerMachine
Simulator";

timer1.Enabled = true;
timer1.Interval = 5000;
timer1.Start();
timer1.Tick += new EventHandler(timer1_Tick);
}
private void timer1_Tick(object sender, EventArgs e)
{
txbadminInfo.Clear();
Application.Exit();
}
private void btPrintRport_Click(object sender, EventArgs e)
{
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Sorry this Function Is Temporarily out of Service";
risenewoperation();
}
private void btDisplayReport_Click(object sender, EventArgs e)
{
txbadminInfo.Clear();
txbadminInfo.Text = nnline + " Sorry this Function Is Temporarily out of Service";
risenewoperation();
}
private void btTakeOutofService_Click(object sender, EventArgs e)
{
FRMOutOfService outofservice = new FRMOutOfService(atm,mylog);
outofservice.Show();
outofservice.Show();
}}}
```

8.0 Class Library

Account.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

// Name: Dangelo Trindade Santana | dangelosantana@hotmail.com
// Professor: Lounis Zaidi | lounis.zaidi@collegecdi.ca
// Subject: Project Integration Level-1 PP1E
// Date: JAN-2017

namespace ATMclassLibrary
{
    public abstract class Account
    {
        // PROPERTIES AND FEATURES
        // PIN  NUMBER
        private string _pinnumber;
        public string pinnumber
        {
            get { return _pinnumber; }
            set { _pinnumber = value; }
        }
        // ACCOUNT NUMBER
        private string _acNumber;
        public string acNumber
        {
            get { return _acNumber; }
            set { _acNumber = value; }
        }
        // ACCOUNT BALANCE
        private decimal _acBalance;
        public decimal acBalance
        {
            get { return _acBalance; }
            set { _acBalance = value; }
        }
        // ACCOUNT MAXIMUM WITHDRAW
        private decimal _maxWithdraw;
        public decimal maxWithdraw
        {
            get { return _maxWithdraw; }
            set { _maxWithdraw = value; }
        }
        // ACCOUNT MAXIMUM TRANSFER AMOUNT
        private decimal _maxTransfer;
        public decimal maxtransfer
        {
            get { return _maxTransfer; }
            set { _maxTransfer = value; }
        }
        // CONSTRUCTOR
        public Account()
        { }
public Account(string PIN, string acNo, decimal acBal, decimal acMaxdraw, decimal
maxtransf)
```

```
        {
            pinnumber = PIN;
            acNumber = acNo;
            acBalance = acBal;
            maxWithdraw = acMaxdraw;
            maxtransfer = maxtransf;
        }
        public Account(string PIN, string acNo, decimal acBal)
        {
            pinnumber = PIN;
            acNumber = acNo;
            acBalance = acBal;
            maxWithdraw = 1000;
            maxtransfer = 10000;
        }
        // ***METHODS***
        // WITHDRAW
        public decimal withdraw(decimal wdAmount)
        {
            try
            {
                acBalance -= wdAmount;
            }
            catch (Exception e)
            { throw new Exception(e.Message); }
            return acBalance;
        }
        // DEPOSIT
        public decimal deposit(decimal creditAmount)
        {
            try
            {
                acBalance += creditAmount;
            }
            catch (Exception ex)
            {
                throw new Exception(ex.Message);
            }
            return acBalance;
        }
        // TRANSFER IN
        public decimal transferIN(decimal creditedamount)
        {
            try
            {
                acBalance += creditedamount;
            }
            catch (Exception e)
            {
                throw new Exception(e.Message);
            }
            return acBalance;
        }
        // TRANSFER OUT
        public decimal transferOUT(decimal debitedamount)
        {
            try
            {
                acBalance -= debitedamount;
            }
            catch (Exception e)
            {
                throw new Exception(e.Message);
            }
            return acBalance;
        }
        // GETPIN
        public string GetPin()
        {
            return pinnumber;
        }
        // GET ACCOUNT NUMBER
        public string GetAcNumber()
        {
            return acNumber;
        }
        // GETBALANCE
        public decimal GetBalance()
        {
            return acBalance;
        }
    }
}
```

Saving.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ATMclassLibrary
{
public class Saving : Account
{
// PROPERTIES AND FEATURES
// INTEREST RATE
private const decimal interestRate = 0.365m;//1%(balance*rate/365/100)
// CONSTRUCTORS
public Saving()
{

}
public Saving(string PIN, string acNo, decimal acBal) : base(PIN, acNo, acBal)
{

}
// ***METHODS***
// PAY INTEREST
public decimal payinterest()
{
acBalance = acBalance + interestRate;
return acBalance;
}
}
}
```

Checking.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ATMclassLibrary
{
public class Checking : Account
{
// PROPERTIES AND FEATURES
// BILL FEE
public decimal billfee { get; set; }
// MAXIMUM BILL AMOUNT
private decimal _MaximumBillAmount;
public decimal MaxBillAmount
{
get { return _MaximumBillAmount; }
set { _MaximumBillAmount = value; }
}
// CONSTRUCTORs
public Checking()
{

}
public Checking(string PIN, string acNo, decimal acBal) : base(PIN, acNo, acBal)
{

}
// ***METHODS***
// PAY BILL
public bool paybill(decimal amount)
{
bool Paymentstatus = false;
acBalance = acBalance - amount - 1.25m;
return Paymentstatus;
}
}
}
```

Bank.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ATMclassLibrary
{
public class Bank : Account
{
public const decimal maximumTopUp = 20000.00m;
public const decimal refillamount = 5000;
// CONSTRUCTOR
public Bank()
{

}
public Bank(string PIN, string acNo, decimal acBal) : base(PIN, acNo, acBal)
{

}
// *** METHODS ***
// REFILL ATM
public void refillATM()
{
try
{
while (acBalance+refillamount < maximumTopUp)
{
acBalance += refillamount;
}
}
catch (Exception ex)
{
throw new Exception(ex.Message);
}
}
}
}
```

CheckingsAccounts

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;

namespace ATMclassLibrary
{
public class CheckingAccounts : CollectionBase
{
public void AddChecking(Checking Chck)
{
List.Add(Chck);
}
public Checking this[int i]
{
get
{
return (Checking)List[i];
}
set
{
List[i] = value;
}
}
}
}
```

SavingAccounts.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;

namespace ATMclassLibrary
{
    public class SavingAccounts : CollectionBase
    {
        public void addSavin(Saving sv)
        {
            List.Add(sv);
        }
        public Saving this[int index]
        {
            get
            {
                return (Saving)this[index];
            }
            set
            {
                List[index] = value;
            }
        }
    }
}
```

Customer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ATMclassLibrary
{
    public class Customer
    {
        // PROPERTIES AND FEATURES
        // FIRST NAME
        private string _Name;
        public string Name
        {
            get { return _Name; }
            set { _Name = value; }
        }
        // PIN NUMBER
        private string _pinNumber;
        public string pinNumber
        {
            get { return _pinNumber; }
            set { _pinNumber = value; }
        }
        // CONSTRUCTOR
        public Customer()
        {

        }
        public Customer(string name, string PIN)
        {
            Name = name;
            pinNumber = PIN;
        }
        // GET NAME
        public string GetName()
        {
            return Name;
        }
        // GET PIN
        public string GetPIN()
        {
            return pinNumber;
        }
    }
}
```

Customers.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;

namespace ATMclassLibrary
{
    public class Customers : CollectionBase
    {
        public void addCustomer(Customer customer)
        {
            List.Add(customer);
        }
        public Customer this[int i]
        {
            get
            {
                return (Customer)List[i];
            }
            set
            {
                List[i] = value;
            }
        }
    }
}
```

atmManager.cs

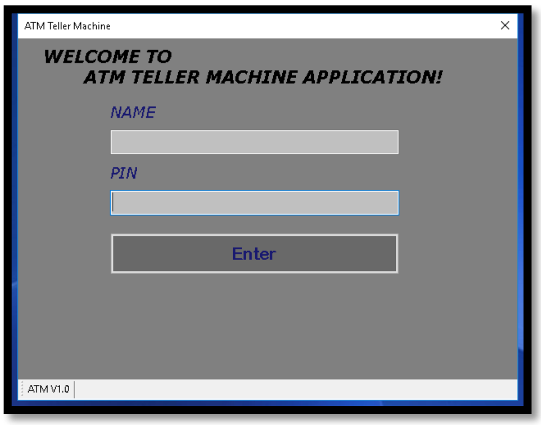
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

// Name: Dangelo Trindade Santana | dangelosantana@hotmail.com
// Professor: Lounis Zaidi | lounis.zaidi@collegecdi.ca
// Subject: Project Integration Level-1 PP1E
// Date: JAN-2017

namespace ATMclassLibrary
{
public class atmManager
{
// BANK DEPLOYMENT
public CheckingAccounts checkingAccounts = new CheckingAccounts();
public SavingAccounts savingAccounts = new SavingAccounts();
public Customers customers = new Customers();
string currentPath = (Environment.CurrentDirectory);
public Bank bank;

// ACCOUNT
public Checking checkingAccount = new Checking();
public Saving savingAccount = new Saving();
// CUSTOMER DEPLOYMENT
public Customer customer = new Customer();
// public Customers customersCollection = new Customers();
// LOADING BANK INFO FILES
public bool loadBank()
{
bool precessSucess = false;
string PIN;
string acNo;
decimal acBal;
try
{
string[] entries;
string AccountsFile = (currentPath + "\\Accounts.txt");
StreamReader streamReader = new StreamReader(AccountsFile);
string line;
line = streamReader.ReadLine();

while (line != null)
{
entries = line.Split(',');
if (entries[0] == "B")
{
PIN = (entries[1]).ToString();
acNo = (entries[2]).ToString();
acBal = Convert.ToDecimal(entries[3]);
bank = new Bank(PIN, acNo, acBal);
precessSucess = true;
}
else if (entries[0] == "C")
{
PIN = (entries[1]).ToString();
acNo = (entries[2]).ToString();
acBal = Convert.ToDecimal(entries[3]);
Checking checking = new Checking(PIN, acNo, acBal);
checkingAccounts.AddChecking(checking);
}
else if (entries[0] == "S")
{
PIN = (entries[1]).ToString();
acNo = (entries[2]).ToString();
acBal = Convert.ToDecimal(entries[3]);
Saving saving = new Saving(PIN, acNo, acBal);
savingAccounts.addSavin(saving);
}
line = streamReader.ReadLine();
}
streamReader.Close();
precessSucess = true;
}
catch (Exception e)
{
throw new Exception(e.Message);
}
return precessSucess;
}
// /READ CUSTOMERS (LOADING CUSTOMER INFO FILE )
public bool loadCustomer()
{
string name;
string Pin;
string[] entries;
string CustomerFile = (currentPath + "\\Customers.txt");
StreamReader customerReader = new StreamReader(CustomerFile);
string line;
line = customerReader.ReadLine();
bool precessSucess = false;
try
{
while (line != null)
{
entries = line.Split(',');
name = entries[0].ToString();
Pin = entries[1].ToString();
Customer cu = new Customer(name, Pin);
customers.addCustomer(cu);
line = customerReader.ReadLine();
}
customerReader.Close();
precessSucess = true;
}
catch (Exception e)
{
throw new Exception(e.Message);
}
return precessSucess;
}
// VALIDATE USER
public bool validateUser(string name, string pin)
{
```

```
bool userOk = false;
try
{
foreach (Customer Cust in customers)
{
if (Cust.Name == name)
{
if (Cust.pinNumber == pin)
{
userOk = true;
customer = Cust;
loadsaving(Cust.pinNumber);
loadchecking(Cust.pinNumber);
break;
}
}
}
}
catch (Exception e)
{
throw new Exception(e.Message);
}
return userOk;
}
// READ ACCOUNTS CURRENT (LOAD CHECKING / LOAD SAVING)
public bool loadsaving(string cuPin)
{
bool precessSucess = false;
foreach (Saving sv in savingAccounts)
{
if (cuPin == sv.pinnumber)
{
savingAccount = sv;
precessSucess = true;
}
}
return precessSucess;
}
public bool loadchecking(string cuPin)
{
bool precessSucess = false;
foreach (Checking chkacc in checkingAccounts)
{
if (cuPin == chkacc.pinnumber)
{
checkingAccount = chkacc;
precessSucess = true;
}
}
return precessSucess;
}
//WITHDRAW CHECKING
public bool ChekingWithdraw(string pin, decimal amount)
{
bool ProcessPermission = false;
decimal tencondiction = amount % 10;
if (amount < 10000 && tencondiction == 0)
{
ProcessPermission = true;
checkingAccount.withdraw(amount);
```

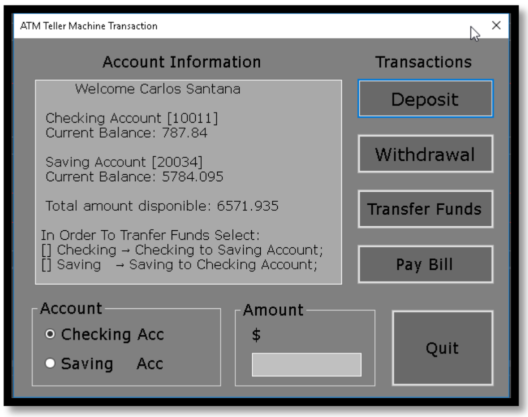
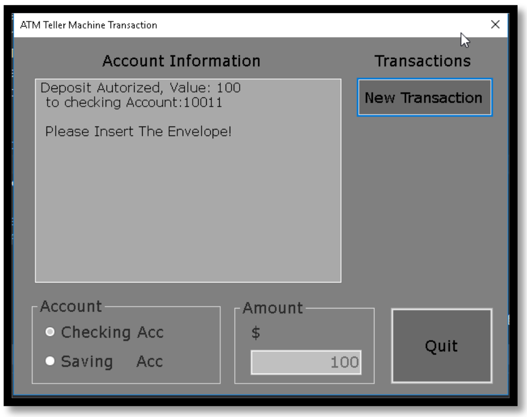
```
}
return ProcessPermission;
}
//WITHDRAW SAVINGS
public bool savingWithdraw(string pin, decimal amount)
{
bool ProcessPermission = false;
decimal tencondiction = amount % 10;
if (amount < 10000 && tencondiction == 0)
{
ProcessPermission = true;
savingAccount.withdraw(amount);
}
return ProcessPermission;
}
//DEPOSIT CHECKING
public bool depositChecking(string pin, decimal amount)
{
bool processStatus = false;
if (pin == checkingAccount.pinnumber)
{
checkingAccount.deposit(amount);
processStatus = true;
}
return processStatus;
}
//DEPOSIT SAVINGS
public bool depositsaving(string pin, decimal amount)
{
bool processStatus = false;
if (pin == savingAccount.pinnumber)
{
savingAccount.deposit(amount);
processStatus = true;
}
return processStatus;
}
//PAY BILL
public bool paybill(string pin, decimal amount)
{
bool processok = false;
decimal checkbalance = checkingAccount.acBalance - amount;
if (pin == checkingAccount.pinnumber && checkbalance >= 0)
{
checkingAccount.paybill(amount);
processok = true;
}
return processok;
}
//TRANSFER FUNDS
public bool transferFunds(string pin, decimal amount, string TransferType)
{
bool processStatus = false;
if (pin == customer.pinNumber)
{
decimal chekingenough = checkingAccount.acBalance - amount;
decimal savingenough = savingAccount.acBalance - amount;
if (TransferType == "CS" && chekingenough >= 0)//checking to save
{
checkingAccount.transferOUT(amount);
savingAccount.transferIN(amount);
processStatus = true;
}
else if (TransferType == "SC" && savingenough >= 0)// saving to checking
{
savingAccount.transferOUT(amount);
checkingAccount.transferIN(amount);
processStatus = true;
}}
return processStatus;
}
//DISPLAY ACCOUNT BALANCE
public decimal DisplayCheckingBalance()
{
decimal balance = checkingAccount.acBalance;
return balance;
}
public decimal DisplaySavingBalance()
{
decimal balance = savingAccount.acBalance;
return balance;
}
//WRITEACCOUNTS
public void WriteAccount()
{
try
{
string checkingInfo;
string bankInfo;
string savingInfo;
string AccountsFile = (currentPath + "\\Accounts.txt");
StreamWriter streamWriter = new StreamWriter(AccountsFile);

bankInfo = "B," + bank.pinnumber.ToString() + "," + bank.acNumber.ToString() + "," +
bank.acBalance.ToString();
streamWriter.WriteLine(bankInfo);
foreach (Checking check in checkingAccounts)
{
checkingInfo = "C," + check.pinnumber.ToString() + ","+check.acNumber.ToString() + "," +
check.acBalance.ToString();
streamWriter.WriteLine(checkingInfo);
}
foreach (Saving sv in savingAccounts)
{
savingInfo = "S," + sv.pinnumber.ToString() + "," + sv.acNumber.ToString() + "," +
sv.acBalance.ToString();
streamWriter.WriteLine(savingInfo);
}
streamWriter.Close();
}
catch (Exception e)
{
throw new Exception(e.Message);
}}}}
```

9.0 ATM Windows Form and Manual

9.1 Login Form Design

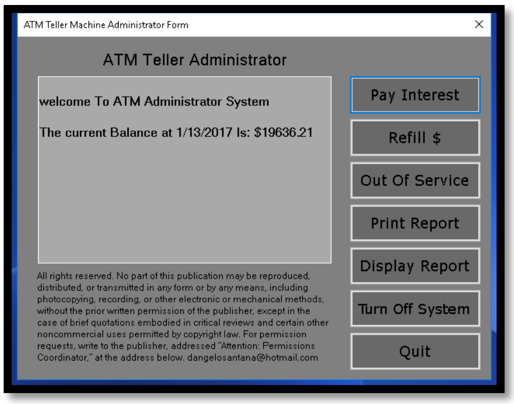
Login Form Design	
 ATM Teller Machine WELCOME TO ATM TELLER MACHINE APPLICATION! NAME PIN Enter ATM V1.0	First form presented, destined to load data of the user and to initiate the transactions of the system.

9.2 Customer Transaction Form Design

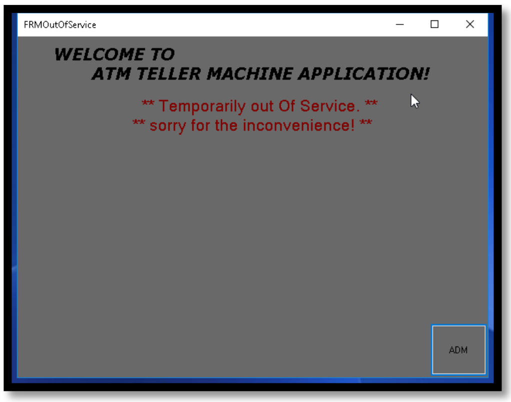
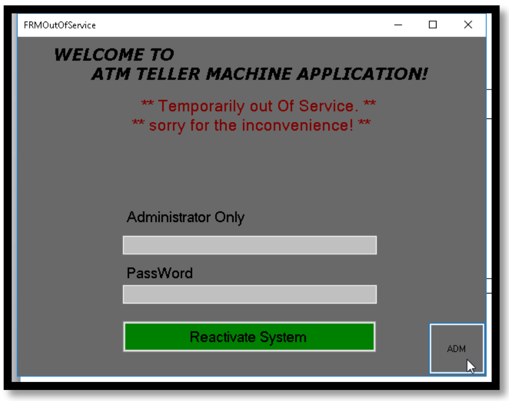
Customer Transaction Form Design	
 ATM Teller Machine Transaction Account Information Welcome Carlos Santana Checking Account [10011] Current Balance: 787.84 Saving Account [20034] Current Balance: 5784.095 Total amount disponible: 6571.935 In Order To Tranfer Funds Select: [] Checking → Checking to Saving Account; [] Saving → Saving to Checking Account; Transactions: Deposit, Withdrawal, Transfer Funds, Pay Bill Account: Checking Acc, Saving Acc Amount: $ Quit	Standard client form aim give access to the follow transactions: • Deposit; • Withdrawal; • Transfer Funds • Pay Bill; In addition of display account details (saving and checking).
 ATM Teller Machine Transaction Account Information Deposit Autorized, Value: 100 to checking Account:10011 Please Insert The Envelope! Transactions: New Transaction Account: Checking Acc, Saving Acc Amount: $ 100 Quit	After progress any operation, the “New transaction” Button will rise up and a confirmation (or error message) will be displayed in the screen.

CDI College

9.3 Administrator Transaction Form Design

Administrator Transaction Form Design	
 ATM Teller Machine Administrator Form ATM Teller Administrator welcome To ATM Administrator System The current Balance at 1/13/2017 Is: $19636.21 All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed "Attention: Permissions Coordinator," at the address below. dangelosantana@hotmail.com Pay Interest Refill $ Out Of Service Print Report Display Report Turn Off System Quit	Administrator client Form aim give access to particular administrator tasks such as: • Refill; • Out Off Service; • Print Report; • Display Report; • Turn Off System; In addition of display Bank account details as current balance.

9.4 Out Of Service Form Design

Out Of Service Form Design	
 FRMOutOfService WELCOME TO ATM TELLER MACHINE APPLICATION! ** Temporarily out Of Service. ** ** sorry for the inconvenience! ** 	The Out of Service button will initiate take the Machine "out of Service", displaying the present Form.
 FRMOutOfService WELCOME TO ATM TELLER MACHINE APPLICATION! ** Temporarily out Of Service. ** ** sorry for the inconvenience! ** Administrator Only PassWord 	The "ADM" button (inferior right corner) will display temporally (10') the reactivate system container (textbox and button). Reserved to Administrator account only, the present task aim unlock the ATM Machine e send back to normal operation.

9.0 Entry Data Test Sample

Accounts.txt	Customer.txt
Account Type, Pin, Acc. No, Acc. Balance B,0000,00000,19636.21 C,D001,10001,457.98 C,C001,10021,1028.49 C,C002,10031,4.10 C,S001,10011,787.84 C,J001,10005,7210.00 S,D001,20030,22.625 S,C001,20101,779.975 S,C002,20001,55.545 S,S001,20034,5784.095 S,J001,20008,36.095	Name , PPIN Korben Dallas,D001 Jerry Cann,C001 Eric Clapton,C002 Carlos Santana,S001 Elton John,J001

6. Conclusion

Through the integration project, it was possible to practice important learning acquired until the present moment. Also, engage the theory building a programming application closer to the real world with visual Studio and C Sharp console application in additional with programming and logical design, programming fundamentals, system analysis design.

Overall, the utilization of Programming Object Oriented, in addition to better organize the software structure, provides a better reading of them in the self-niche, minimize errors while improving the quality and the speed in debug the software.

The system.io stream reader provides a powerful feature in order to keep some information in the system, using a simple and fast way.

The Windows Form Application provides a robust and Simple implementation front edge appliance and it becomes a great way to build a form to be used by any kind systems.

Programming Object Oriented can be perceived as a potential method in order to reduce spending in the construction and maintenance of the system proposed, rising up the algorithm to a level near to the reality without losing security of data.