*ProgramAsignment3*

**Source code**

*Form1.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Principal;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static ProgramAssignment3.Program;

namespace ProgramAssignment3

{

public partial class Form1 : Form

{

BankAccount bAccount;

public Form1()

{

InitializeComponent();

}

private void loginBtn\_Click(object sender, EventArgs e)

{

if (fNameTb.Text != "" && lNameTb.Text != "" && balanceTb.Text != "")

{

LoginPnl.Visible = false;

AccountPnl.Visible = true;

bAccount = new BankAccount("12345", fNameTb.Text, lNameTb.Text, Convert.ToInt32(balanceTb.Text));

loadMethod();

}

else {

if (fNameTb.Text == "") { fNameTb.BackColor = Color.Red; fNameTb.Focus(); }

if (lNameTb.Text == "") { lNameTb.BackColor = Color.Red; lNameTb.Focus(); }

if (balanceTb.Text == "") { balanceTb.BackColor = Color.Red; balanceTb.Focus(); }

}

}

void loadMethod() {

label5.Text = bAccount.FirstName;

label6.Text = bAccount.LastName;

label7.Text = bAccount.Balance.ToString();

comboBox1.SelectedIndex = 0;

}

private void button1\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex != 1) {

if (bAccount.deposit(Convert.ToInt32(textBox1.Text)) == "1"){

label7.Text = bAccount.Balance.ToString();

textBox2.Text = "Transaction approved.";

}

else {

textBox2.Text = (bAccount.deposit(Convert.ToInt32(textBox1.Text)));

}

}

else

{

if (bAccount.withdraw(Convert.ToInt32(textBox1.Text)) == "1")

{

label7.Text = bAccount.Balance.ToString();

textBox2.Text = "Transaction approved.";

}

else

{

textBox2.Text = (bAccount.withdraw(Convert.ToInt32(textBox1.Text)));

}

}

}

private void textBox1\_Click(object sender, EventArgs e)

{

textBox2.Text = "";

}

private void comboBox1\_Click(object sender, EventArgs e)

{

textBox2.Text = "";

}

private void fNameTb\_Leave(object sender, EventArgs e)

{

if (fNameTb.Text == "")

{

fNameTb.Focus();

//MessageBox.Show("Can't be empty");

fNameTb.BackColor = Color.Red;

}

}

private void lNameTb\_Leave(object sender, EventArgs e)

{

if (lNameTb.Text == "")

{

lNameTb.Focus();

//MessageBox.Show("Can't be empty");

lNameTb.BackColor = Color.Red;

}

}

private void balanceTb\_Leave(object sender, EventArgs e)

{

if (balanceTb.Text == "")

{

balanceTb.Focus();

//MessageBox.Show("Can't be empty");

balanceTb.BackColor = Color.Red;

}

}

private void fNameTb\_KeyPress(object sender, KeyPressEventArgs e)

{

fNameTb.BackColor = Color.White;

}

private void lNameTb\_KeyPress(object sender, KeyPressEventArgs e)

{

lNameTb.BackColor = Color.White;

}

private void balanceTb\_KeyPress(object sender, KeyPressEventArgs e)

{

balanceTb.BackColor = Color.White;

}

}

}

*Program.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Console;

namespace ProgramAssignment3

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

internal class BankAccount

{

private string id; // only set using constructor, provide an accessor method

private string firstName; // need to provide accessor and mutator methods

private string lastName; // need to provide accessor and mutator methods

private double balance; // Only set using constructor, provide an accessor method

private int numTransactions; // Initialized to zero in constructor, provide an accessor method

// Please provide Accessors and/or Mutator methods as well as using at least one Property method (for the fields and rules above)

// Please a few constructors... default constructor as well as a few parameterized Constructors:

/\* Note: if balance is not specified in constructor, it should be initializer to zero

Note: All constructors shall set numTransactions to zero

Constructor example: id, First and Last name, & balance

\*/

public string Id

{

get

{

return id;

}

}

public string FirstName

{

get

{

return firstName;

}

set

{

firstName = value;

}

}

public string LastName

{

get

{

return lastName;

}

set

{

lastName = value;

}

}

public double Balance

{

get

{

return balance;

}

}

public double NumTransactions

{

get

{

return numTransactions;

}

}

public BankAccount()

{

this.numTransactions = 0;

}

public BankAccount(string ID, string fname, string lname, double Balance = 0)

{

this.id = ID;

this.firstName = fname;

this.lastName = lname;

this.balance = Balance;

this.numTransactions = 0;

}

// Additional methods

// Deposit - Adds amount to balance. Also counts as 1 transaction.

public string deposit(double amount) // Note verify amount is positive and between 1..1000 (inclusive)

{

if (amount >= Constants.MIN && amount <= Constants.MAX)

{

this.balance += amount;

this.numTransactions++;

return "1";

}

else

return "The amount exceeds the maximum possible deposit!";

}

// Subtracts amount from balance if user has enough money. Counts as 1 transaction.

public string withdraw(double amount)

{

if (this.balance >= amount)

{

this.balance -= amount;

this.numTransactions++;

return "1";

}

else

return "You have less money then you want withdraw :(";

}

public string ReturnFullName()

{

return (firstName + " " + lastName);

}

public string toString() // This will return a string containing the account info: F and L name/ID and balance and num transactions.

{

return "Account Number: " + id +

"\nName: " + ReturnFullName() +

"\nBalance: " + balance + "$" +

"\nNumber of transactions: " + numTransactions;

}

}

}

}

*Constants.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ProgramAssignment2

{

public class Constants

{

public const int MAX = 1000;

public const int MIN = 1;

}

}

**Outputs**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**