**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

**«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**ЗВІТ**

про виконання лабораторної роботи №1 за темою

«Основи розробки додатків у .NET»

з дисципліни «Стек технологій»

Виконав студент групи КН-322б Загорський Артем

Перевірила доцентка кафедри УПІТ Олександр Олександрович Лисенко

Харків

НТУ «ХПІ»

2024

**Лабораторна робота №1**

**Мета:** Навчитися створювати консольні та віконні додатки, ознайомтись з використання делегатів та подій у мові програмування C#.

**ЗАВДАННЯ**

1. Опрацювати теорію (див. додаток А)

2. Написати програму для моделювання роботи банкомату.

Рішення має складатися з трьох проектів:

* бібліотека класів;
* віконний проект;
* консольний проект.

У бібліотеці класів реалізуйте загальні типи даних, які будуть використані у віконному та консольному проектах. У віконному додатку розробіть інтерфейс, який наближений до інтерфейсу традиційного банкомату. Консольний додаток повинен забезпечити ті ж самі можливості, що має віконний додаток, але за допомогою консольного меню.

Слід передбачити наступні можливості:

* аутентифікація (перевірка введення номеру картки та пін-коду);
* перегляд балансу на картці;
* перегляд історіі операцій з фільтруванням по проміжку часу (за поточний день, за поточний тіждень, за поточний місяц)
* зняття коштів;
* зарахування коштів на картку;
* перерахування коштів на картку із заданим номером;
* перегляд найближчих банкоматів;

Повинна бути передбачена можливість додавання власних обробників подій на виконання операцій аутентифікації, перегляду балансу, зняття коштів та перерахування коштів на іншу картку.

**Хід роботи**

Спочатку було створено бібліотеку класів, яка містить три основні компоненти: **Account**, **AutomatedTellerMachine** та **Bank**. Крім того, розроблено клас **Transaction** та два типи перерахувань: **TransactionType** і **TransactionPeriod**.

**Код классів**

Account

namespace BankLibrary

{

public class Account

{

public string cardNumber;

public int pin;

public string name;

public float money;

public Account(string \_cardNumber,int \_pin, string \_name, float \_money)

{

cardNumber = \_cardNumber;

pin = \_pin;

name = \_name;

money = \_money;

}

public void Withdraw(float amount)

{

if(money >= amount)

{

money -= amount;

}

}

public void Topup(float amount)

{

if (amount > 0)

{

money += amount;

}

}

public float getCurrentMoneyNumber()

{

return money;

}

public void setMoneyNumber(float newAmount)

{

money = newAmount;

}

public bool pinChecker(int \_pin)

{

return pin == \_pin;

}

public bool cardNumberCheker(string \_cardNumber)

{

return cardNumber == \_cardNumber;

}

}

}

AutomatedTellerMachine

namespace BankLibrary

{

public class AutomatedTellerMachine

{

public float moneyCount;

public int id;

public Bank bank;

public Account currentAccount;

public List<Transaction> transactions = new List<Transaction> ();

public AutomatedTellerMachine(float moneyCount, int id, Bank bank)

{

this.moneyCount = moneyCount;

this.id = id;

this.bank = bank;

}

public float getCurrentBallance()

{

if (currentAccount != null)

return currentAccount.getCurrentMoneyNumber();

return -1;

}

//Todo add all transitions to Transitions array

public void WithdrawOperation(float amount, int pin)

{

if (currentAccount != null && currentAccount.pinChecker(pin) && amount <= moneyCount)

currentAccount.Withdraw(amount);

AddNewTransaction(amount, TransactionType.Withdraw);

}

public void TopUpOperation(float amount, int pin)

{

if (currentAccount != null && currentAccount.pinChecker(pin))

currentAccount.Topup(amount);

AddNewTransaction(amount, TransactionType.TopUp);

}

public void TransferOperation(float amount, int pin, string recieverCardNumber)

{

if (currentAccount != null && currentAccount.pinChecker(pin) && amount > 0)

{

currentAccount.Withdraw(amount);

bank.GetAccountByCardNumber(recieverCardNumber).Topup(amount);

AddNewTransaction(amount, TransactionType.Transfer, recieverCardNumber);

}

}

public List<Transaction> GetTransactionHistory(TransactionPeriod period)

{

List<Transaction> filteredTransactions = new List<Transaction>();

DateTime now = DateTime.Now;

string dateFormat = "dd/MM/yyyy HH:mm:ss";

foreach (var transaction in transactions)

{

DateTime transactionDate;

if (DateTime.TryParseExact(transaction.date, dateFormat, null, System.Globalization.DateTimeStyles.None, out transactionDate))

{

switch (period)

{

case TransactionPeriod.day:

if (transactionDate.Day == now.Day)

{

filteredTransactions.Add(transaction);

}

break;

case TransactionPeriod.week:

var currentWeekStart = now.AddDays(-(int)now.DayOfWeek);

var currentWeekEnd = currentWeekStart.AddDays(7);

if (transactionDate >= currentWeekStart && transactionDate < currentWeekEnd)

{

filteredTransactions.Add(transaction);

}

break;

case TransactionPeriod.month:

if (transactionDate.Year == now.Year && transactionDate.Month == now.Month)

{

filteredTransactions.Add(transaction);

}

break;

}

}

}

return filteredTransactions;

}

public void AddNewTransaction(float \_amount, TransactionType \_type, string cardNumber2 = "")

{

Transaction tr = new Transaction();

tr.amount = \_amount;

tr.type = \_type;

tr.cardNumber = currentAccount.cardNumber;

DateTime lastTransactionDate = transactions.Count > 0

? DateTime.ParseExact(transactions.Last().date, "dd/MM/yyyy HH:mm:ss", null)

: DateTime.Now;

Random rand = new Random();

//int randomDays = rand.Next(0, 21);

int randomHours = rand.Next(1, 12);

int randomMinutes = rand.Next(0, 60);

DateTime newTransactionDate = lastTransactionDate;

if (transactions.Count > 0)

{

newTransactionDate = lastTransactionDate.AddHours(randomHours).AddMinutes(randomMinutes);//.AddDays(randomDays)

}

tr.date = newTransactionDate.ToString("dd/MM/yyyy HH:mm:ss");

switch (\_type)

{

case TransactionType.Withdraw:

tr.cardNumber = currentAccount.cardNumber;

break;

case TransactionType.Transfer:

tr.cardNumber = currentAccount.cardNumber;

tr.cardNumber2 = cardNumber2;

break;

case TransactionType.TopUp:

tr.cardNumber = currentAccount.cardNumber;

break;

}

transactions.Add(tr);

}

}

}

Bank

namespace BankLibrary

{

public class Bank

{

public string name;

public List<AutomatedTellerMachine> machines = new List<AutomatedTellerMachine>();

public List<Account> accounts = new List<Account>();

public Account GetAccountByCardNumber(string cardNumber)

{

if (accounts == null)

return null;

for(int i = 0; i < accounts.Count; i++)

{

if(accounts[i].cardNumberCheker(cardNumber))

{

return accounts[i];

}

}

return null;

}

public void AddNewAccount(string \_cardNumber, int \_pin, string \_name, float \_money)

{

Account newAccount = new Account(\_cardNumber, \_pin, \_name, \_money);

accounts.Add(newAccount);

}

public void AddNewATM(float moneyCount, int id, Bank bank)

{

AutomatedTellerMachine machine = new AutomatedTellerMachine(moneyCount, id, bank);

machines.Add(machine);

}

public bool isCardExist(string \_cardNumber)

{

for(int i = 0; i < accounts.Count; i++)

{

if (accounts[i].cardNumberCheker(\_cardNumber))

{

return true;

}

}

return false;

}

}

}

Transaction

public class Transaction

{

public float amount;

public TransactionType type;

public string date;

public string cardNumber;

public string cardNumber2;

}

TransactionType

public enum TransactionType

{

Withdraw,

Transfer,

TopUp

}

TransactionPeriod

public enum TransactionPeriod

{

day,

week,

month

}

Далі було розроблено віконний додаток для банкомату. Створено головну форму **Form1**, а також форму **CardInput** для авторизації користувача. Для операцій зняття готівки та поповнення рахунку створено окрему форму **TransactionForm**. Для здійснення переказів розроблено додаткову форму **TransactionForm2**. Нарешті, було створено форму **History** для перегляду історії транзакцій. Нижче представлені ілюстрації інтерфейсу та програмний код додатку.

**Form1**

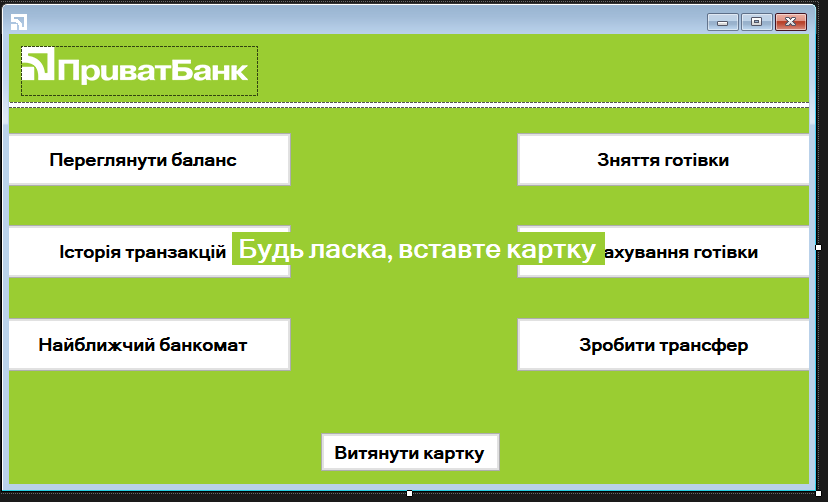
****

Рисунок 1 головна форма банкомату

Код

using BankLibrary;

namespace Lab\_1\_Banking\_App

{

public delegate void GetCardAccount(string cardNumber);

public partial class Form1 : Form

{

public Bank bank = new Bank();

public AutomatedTellerMachine currentATM;

public CardInput cardInputForm = new CardInput();

public TransactionForm transactionForm = new TransactionForm();

public TransitionForm2 transactionForm2 = new TransitionForm2();

public History historyForm = new History();

public Form1()

{

bank.AddNewATM(1200000f, 124, bank);

currentATM = bank.machines[0];

bank.AddNewAccount("1111222233334444", 1234, "John", 12000);

bank.AddNewAccount("2222333344445555", 5555, "Steven", 350);

bank.AddNewAccount("2222888877776666", 0000, "Bob", 1500000);

cardInputForm.bank = bank;

cardInputForm.cardHandler = SetCurrentAccountToATM;

InitializeComponent();

if (currentATM.currentAccount == null)

{

SetActiveMainMenu(false);

SetActivePutTheCardScreen(true);

cardInputForm.Show();

}

}

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

{

MessageBox.Show("На карті " + currentATM.currentAccount.getCurrentMoneyNumber() + " деняг!", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

MessageBox.Show("Найближчий банкомат у 45ти метрів", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

public void SetActiveMainMenu(bool active)

{

this.watchBalancebutton.Visible = active;

this.transactionHistorybutton.Visible = active;

this.nearestATMButton.Visible = active;

this.withdrawButton.Visible = active;

this.putMoneybutton.Visible = active;

this.makeTransactionbutton.Visible = active;

this.signOutButton.Visible = active;

}

public void SetActivePutTheCardScreen(bool active)

{

this.label1.Visible = active;

}

public void SetCurrentAccountToATM(string cardNumber)

{

currentATM.currentAccount = bank.GetAccountByCardNumber(cardNumber);

SetActiveMainMenu(true);

SetActivePutTheCardScreen(false);

cardInputForm.Hide();

}

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

{

currentATM.currentAccount = null;

SetActiveMainMenu(false);

SetActivePutTheCardScreen(true);

cardInputForm.Show();

}

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

{

transactionForm.machine = currentATM;

transactionForm.Initialize(true);

transactionForm.Show();

}

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

{

transactionForm.machine = currentATM;

transactionForm.Initialize(false);

transactionForm.Show();

}

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

{

transactionForm2.machine = currentATM;

transactionForm2.Initialize();

transactionForm2.Show();

}

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

{

historyForm.Form\_Load(currentATM);

historyForm.Show();

}

}

}

**CardInput**

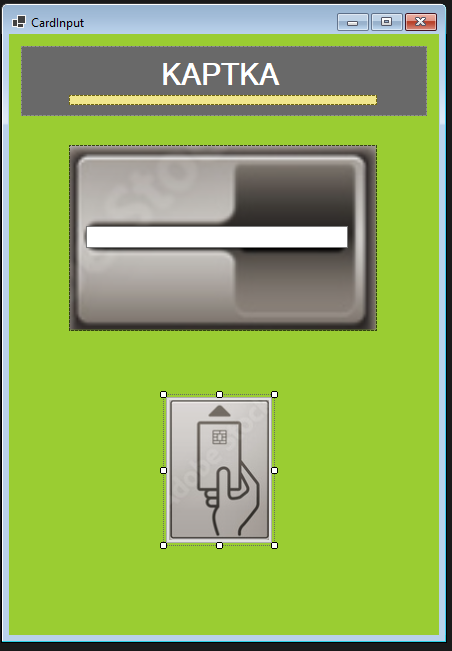


Рисунок 2 форма авторизації користувача

Код

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 BankLibrary;

namespace Lab\_1\_Banking\_App

{

public partial class CardInput : Form

{

public Bank bank;

public GetCardAccount cardHandler;

public CardInput()

{

InitializeComponent();

}

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

{

string cardNumber = textBox1.Text;

if(bank.isCardExist(cardNumber))

{

cardHandler(cardNumber);

}

else

{

MessageBox.Show("Такої катрки не існує", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**TransactionForm**

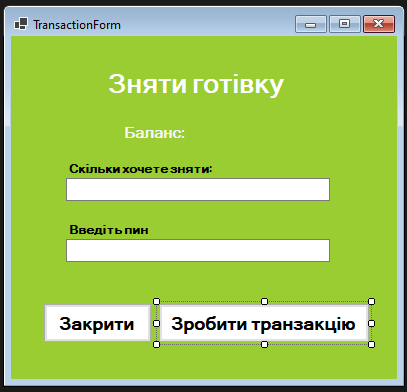


Рисунок 3 форма зняття готівки та поповнення картки

Код

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 BankLibrary;

namespace Lab\_1\_Banking\_App

{

public partial class TransactionForm : Form

{

public AutomatedTellerMachine machine;

public bool withdraw = true;

public TransactionForm()

{

InitializeComponent();

}

public void Initialize(bool \_withdraw)

{

if (machine != null)

{

withdraw = \_withdraw;

if(withdraw)

{

labelTitle.Text = "Зняти готівку";

labelTitle.Location = new System.Drawing.Point(91, 31);

label1.Text = "Скільки хочете зняти:";

}

else

{

labelTitle.Text = "Зарахувати готівку";

labelTitle.Location = new System.Drawing.Point(52, 31);

label1.Text = "Скільки хочете зарахувати:";

}

textBox1.Text = "";

textBox2.Text = "";

label2.Text = "Баланс: " + machine.currentAccount.money.ToString();

}

}

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

{

if (machine.currentAccount == null) return;

float money = float.Parse(textBox1.Text, System.Globalization.CultureInfo.InvariantCulture);

int pin = int.Parse(textBox2.Text);

if ((withdraw && money > 0 && money <= machine.currentAccount.money && money <= machine.moneyCount) || (!withdraw && money > 0) && machine.currentAccount.pinChecker(pin))

{

if(withdraw)

{

machine.WithdrawOperation(money, pin);

}

else

{

machine.TopUpOperation(money, pin);

}

MessageBox.Show("Операція завершена!) На балансі: " + machine.currentAccount.money.ToString() + "деняг!", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Hide();

}

else

{

MessageBox.Show("Помилка", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Hide();

}

private void TransactionForm\_Load(object sender, EventArgs e)

{

}

}

}

**TransactionForm2**

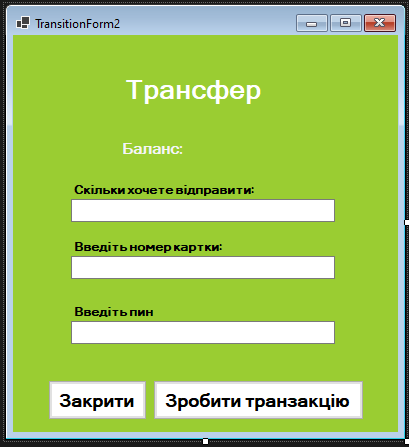
****

Рисунок 4 форма для трансферу грошей

Код

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 BankLibrary;

namespace Lab\_1\_Banking\_App

{

public partial class TransitionForm2 : Form

{

public AutomatedTellerMachine machine;

public TransitionForm2()

{

InitializeComponent();

}

public void Initialize()

{

if (machine != null)

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

label2.Text = "Баланс: " + machine.currentAccount.money.ToString();

}

}

private void TransitionForm2\_Load(object sender, EventArgs e)

{

}

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

{

if (machine.currentAccount == null) return;

float money = float.Parse(textBox1.Text, System.Globalization.CultureInfo.InvariantCulture);

int pin = int.Parse(textBox2.Text);

string reciever = textBox3.Text;

if(!machine.bank.isCardExist(reciever)) MessageBox.Show("Помилка", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

if ( money > 0 && money <= machine.currentAccount.money && machine.currentAccount.pinChecker(pin))

{

machine.TransferOperation(money, pin, reciever);

MessageBox.Show("Операція завершена!) На балансі: " + machine.currentAccount.money.ToString() + "деняг!", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Hide();

}

else

{

MessageBox.Show("Помилка", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Hide();

}

}

}

**History**

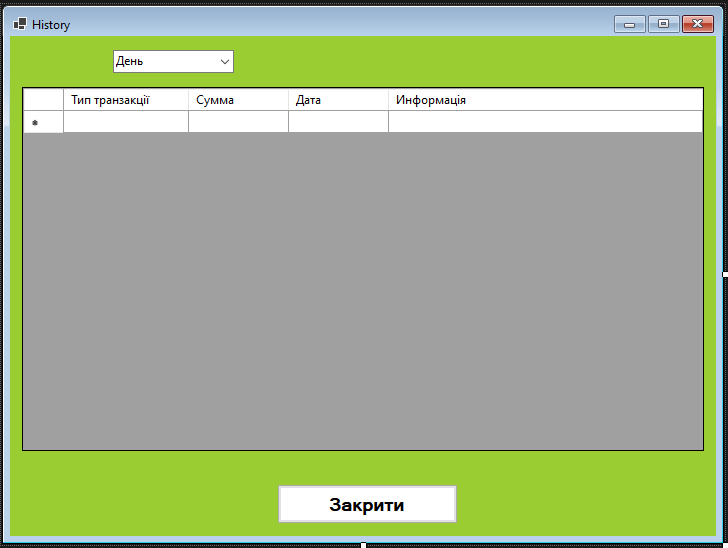


Рисунок 5 форма для перегляду транзакцій банкомату

Код

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 BankLibrary;

namespace Lab\_1\_Banking\_App

{

public partial class History : Form

{

public AutomatedTellerMachine currentATM;

public History()

{

InitializeComponent();

}

public void Form\_Load(AutomatedTellerMachine \_currentATM)

{

currentATM = \_currentATM;

DisplayTransactions(currentATM.GetTransactionHistory(TransactionPeriod.day));

}

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

{

this.Hide();

}

private void DisplayTransactions(List<Transaction> transactions)

{

transactionsDataGridView.Rows.Clear();

foreach (var transaction in transactions)

{

string[] row = new string[] {

transaction.type.ToString(),

transaction.amount.ToString(),

transaction.date,

transaction.type == TransactionType.Transfer ? "Відправник " + transaction.cardNumber + "\nОтримувач " + transaction.cardNumber2 : "Номер аккаунта " + transaction.cardNumber

};

transactionsDataGridView.Rows.Add(row);

}

}

private void History\_Load(object sender, EventArgs e)

{

}

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

{

if(comboBox1.SelectedIndex == 0)

{

DisplayTransactions(currentATM.GetTransactionHistory(TransactionPeriod.day));

}

else if(comboBox1.SelectedIndex == 1)

{

DisplayTransactions(currentATM.GetTransactionHistory(TransactionPeriod.week));

}

else

{

DisplayTransactions(currentATM.GetTransactionHistory(TransactionPeriod.month));

}

}

}

}

Після цього було розроблено консольний додаток для банкомату. Нижче представлено програмний код додатку та результати його роботи.

Код

using System;

using BankLibrary;

public class Program

{

public static void Main(string[] args)

{

Bank bank = new Bank();

AutomatedTellerMachine currentATM = new AutomatedTellerMachine(100000f, 124, bank);

Source source = new Source();

bank.AddNewAccount("1111222233334444", 1234, "John", 12000);

bank.AddNewAccount("2222333344445555", 5555, "Steven", 350);

bank.AddNewAccount("2222888877776666", 0000, "Bob", 1500000);

for (;;)

{

if (currentATM.currentAccount == null)

{

Console.WriteLine("Ви не авторизованi!");

Console.WriteLine("Ведiть номер картки: ");

string cardNumber = Console.ReadLine();

if(bank.isCardExist(cardNumber))

{

currentATM.currentAccount = bank.GetAccountByCardNumber(cardNumber);

}

else

{

Console.WriteLine("Помилка!");

continue;

}

}

switch (source.Menu())

{

case 0:

source.SetError();

break;

case 1:

Console.WriteLine("На балансi " + currentATM.currentAccount.money + " деняг!\n");

break;

case 2:

if(currentATM.transactions.Count == 0)

{

Console.WriteLine("Транзакцiй немає:(");

continue;

}

Console.WriteLine("За який перiод транзакцiї(день, тиждень, мiсяць): ");

string period = Console.ReadLine();

if(period == "день")

{

source.DriteAllTransactions(currentATM.GetTransactionHistory(TransactionPeriod.day));

}

else if(period == "тиждень")

{

source.DriteAllTransactions(currentATM.GetTransactionHistory(TransactionPeriod.week));

}

else if(period == "мiсяць")

{

source.DriteAllTransactions(currentATM.GetTransactionHistory(TransactionPeriod.month));

}

else

{

source.SetError();

}

break;

case 3:

Console.WriteLine("Найближчий банкомат у 45ти метрiв.\n");

break;

case 4:

for(; ; )

{

Console.WriteLine("Якщо хочете закінчити операцію введіть - стоп");

Console.WriteLine("Введіть сумму: ");

string moneyToTake = Console.ReadLine();

if (moneyToTake == "стоп") break;

Console.WriteLine("Введіть пін: ");

string pinString = Console.ReadLine();

if (pinString == "стоп") break;

float money = float.Parse(moneyToTake, System.Globalization.CultureInfo.InvariantCulture);

int pin = int.Parse(pinString);

if ((money > 0 && money <= currentATM.currentAccount.money && money <= currentATM.moneyCount) && currentATM.currentAccount.pinChecker(pin))

{

currentATM.WithdrawOperation(money, pin);

Console.WriteLine("Операція завершена!) На балансі: " + currentATM.currentAccount.money.ToString() + " деняг!\n");

break;

}

else

{

Console.WriteLine("Помилка!!!\n");

}

}

break;

case 5:

for (; ; )

{

Console.WriteLine("Якщо хочете закінчити операцію введіть - стоп\n");

Console.WriteLine("Введіть сумму: ");

string moneyToPut = Console.ReadLine();

if (moneyToPut == "стоп") break;

Console.WriteLine("Введіть пін: ");

string pinString = Console.ReadLine();

if (pinString == "стоп") break;

float money = float.Parse(moneyToPut, System.Globalization.CultureInfo.InvariantCulture);

int pin = int.Parse(pinString);

if (money > 0 && currentATM.currentAccount.pinChecker(pin))

{

currentATM.TopUpOperation(money, pin);

Console.WriteLine("Операція завершена!) На балансі: " + currentATM.currentAccount.money.ToString() + " деняг!\n");

break;

}

else

{

Console.WriteLine("Помилка!!!");

}

}

break;

case 6:

for (; ; )

{

Console.WriteLine("Якщо хочете закінчити операцію введіть - стоп\n");

Console.WriteLine("Введіть сумму: ");

string moneyToPut = Console.ReadLine();

if (moneyToPut == "стоп") break;

Console.WriteLine("Введіть номер картки отримувача: ");

string reciever = Console.ReadLine();

if (reciever == "стоп") break;

Console.WriteLine("Введіть пін: ");

string pinString = Console.ReadLine();

if (pinString == "стоп") break;

float money = float.Parse(moneyToPut, System.Globalization.CultureInfo.InvariantCulture);

int pin = int.Parse(pinString);

if (money > 0 && currentATM.currentAccount.pinChecker(pin) && bank.isCardExist(reciever))

{

currentATM.TransferOperation(money, pin, reciever);

Console.WriteLine("Операція завершена!) На балансі: " + currentATM.currentAccount.money.ToString() + " деняг!\n");

break;

}

else

{

Console.WriteLine("Помилка!!!");

}

}

break;

case 7:

currentATM.currentAccount = null;

Console.WriteLine("Ви вийшли з аккаунту.");

break;

}

}

}

}

public class Source

{

public int Menu()

{

int number;

Console.WriteLine("1. Переглянути баланс.");

Console.WriteLine("2. Iсторiя транзацiй.");

Console.WriteLine("3. Найближчий банкомат.");

Console.WriteLine("4. Зняття готiвки.");

Console.WriteLine("5. Зарахування коштiв.");

Console.WriteLine("6. Транфер.");

Console.WriteLine("7. Вихiд.");

Console.Write("\nОберiть пункт меню: ");

if (int.TryParse(Console.ReadLine(), out number) && number > 0 && number <= 7)

{

return number;

}

else

{

return 0;

}

}

public void SetError()

{

Console.WriteLine("ERROR!!");

}

public void DriteAllTransactions(List<Transaction> transactions)

{

foreach (var transaction in transactions)

{

string info = "Тип " + transaction.type.ToString() + " Сума " + transaction.amount.ToString() + " Дата " + transaction.date + " Додаткова iнформацiя " + (transaction.type == TransactionType.Transfer ? "Вiдправник " + transaction.cardNumber + "\nОтримувач " + transaction.cardNumber2 : "Номер аккаунта " + transaction.cardNumber);

Console.WriteLine(info + "\n");

}

}

}

Результат роботи програми

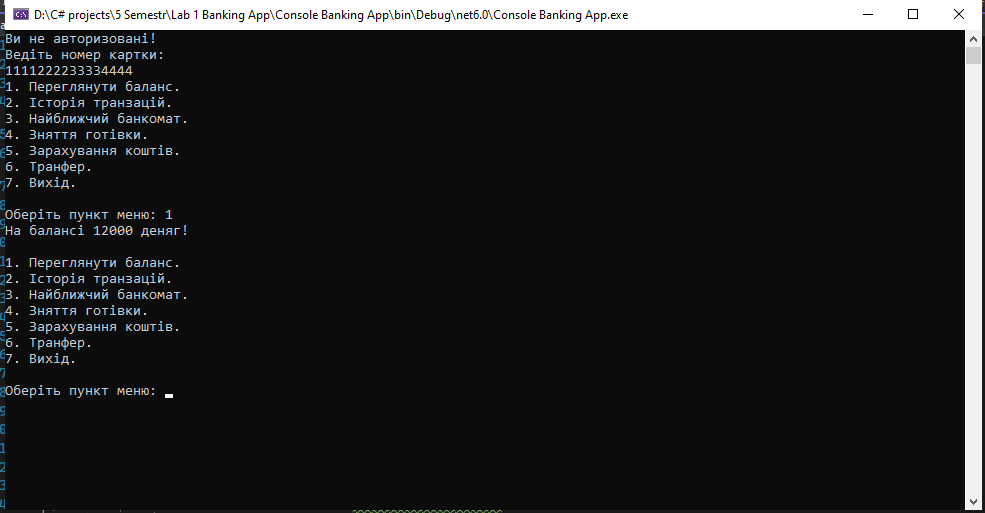


Рисунок 6 Перевірка балансу на аккаунті

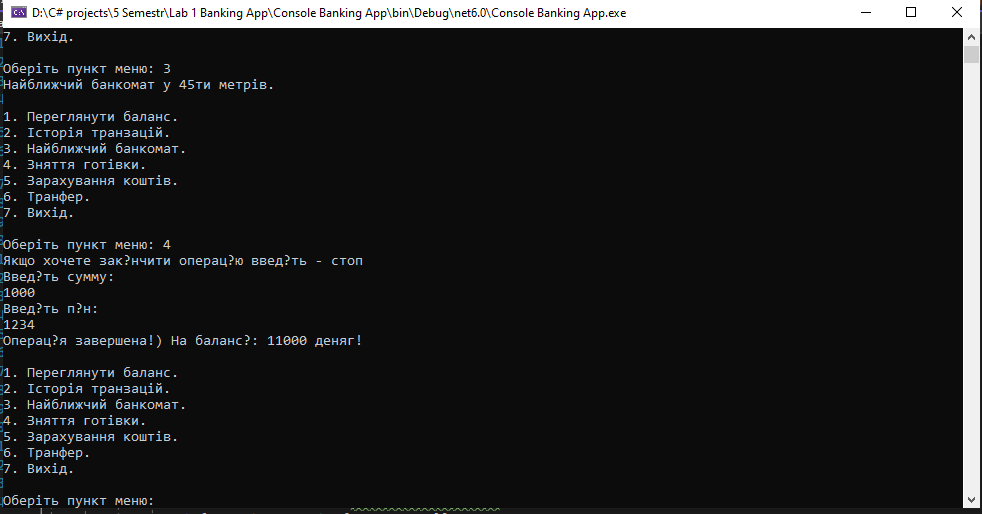


Рисунок 7 Зняття готівки

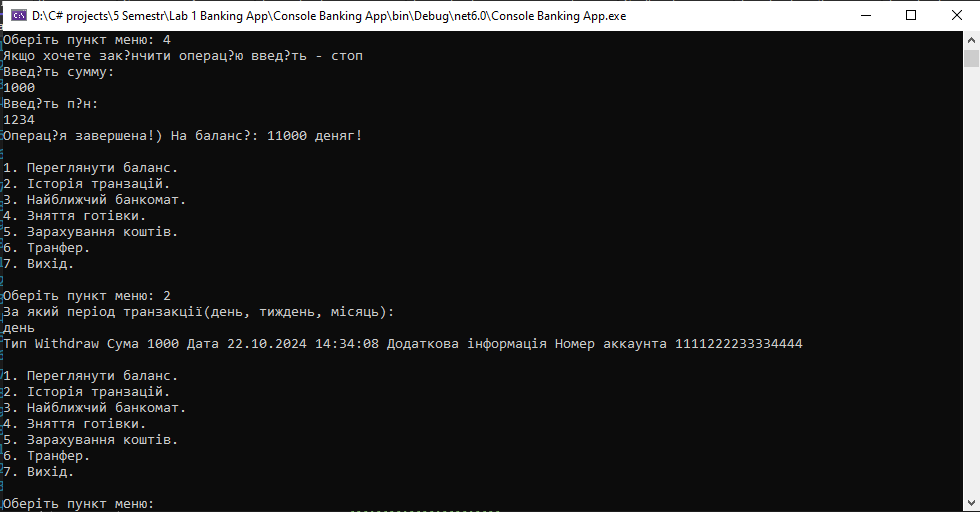


Рисунок 8 Історія транзакцій за день

**ВИСНОВКИ**

У результаті виконання лабораторної роботи я набув навичок створення консольних і віконних додатків на платформі .NET з використанням мови програмування C#. Також я ознайомився з концепціями делегатів та подій, і реалізував їх у проектах для моделювання роботи банкомату. Було створено три проекти: бібліотека класів для загальних типів даних, віконний додаток з інтерфейсом банкомату та консольний додаток, що забезпечує той самий функціонал через меню. Завдання охоплює важливі аспекти розробки програмного забезпечення, включаючи аутентифікацію, роботу з транзакціями та використання подій для обробки операцій.