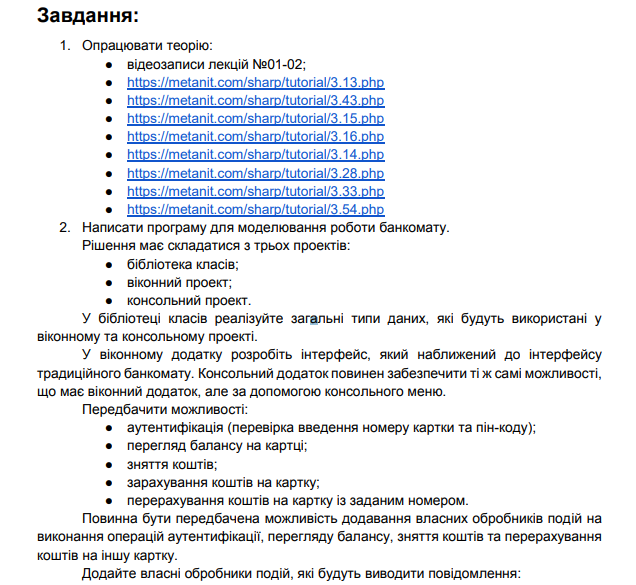
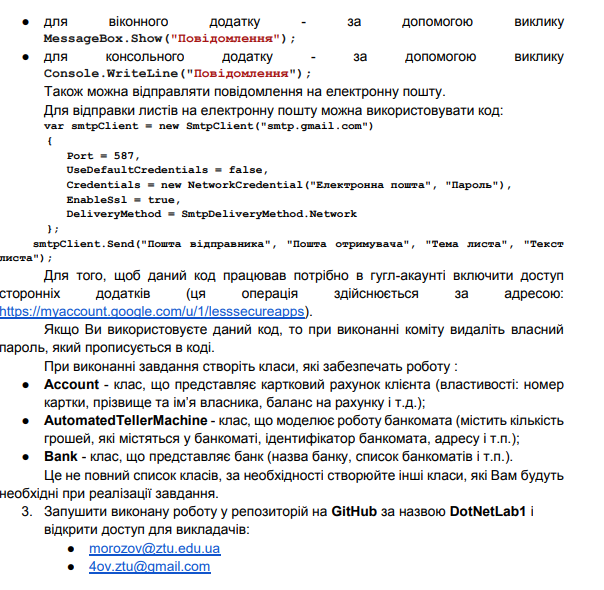
**Лабораторна робота №1**

**Тема:** Використання делегатів та подій у C#

**Мета роботи:** навчитися використовувати оголошувати та використовувати делегати та події у мові програмування C#.





**Бібліотека класів:**

**ObservableObject.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace ATMWPFApp

{

class ObservableObject : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string name = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

}

}

}

**RelayComand.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace ATMWPFApp

{

public class RelayCommand : ICommand

{

private Action<object> execute;

private Func<object, bool> canExecute;

public RelayCommand(Action<object> execute)

{

this.execute = execute;

canExecute = null;

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute)

{

this.execute = execute;

this.canExecute = canExecute;

}

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public bool CanExecute(object parameter)

{

return canExecute == null || CanExecute(parameter);

}

public void Execute(object parameter)

{

execute(parameter);

}

}

}

**Account.cs:**

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

namespace ATMClassLib

{

public delegate void AccountHandler(string message);

public class Account : INotifyPropertyChanged

{

private string \_cardNumber;

private string \_gmailAddress;

private string \_pin;

private string \_name;

private decimal \_balance;

public delegate void PropertyChangedEventHandlerDelegate(object sender, PropertyChangedEventArgs e);

public event PropertyChangedEventHandler PropertyChanged;

public string Pin

{

get { return \_pin; }

set

{

if (\_pin != value)

{

\_pin = value;

OnPropertyChanged(nameof(Pin));

}

}

}

public string GmailAddress

{

get { return \_gmailAddress; }

set

{

if (\_gmailAddress != value)

{

\_gmailAddress = value;

OnPropertyChanged(nameof(GmailAddress));

}

}

}

public string CardNumber

{

get { return \_cardNumber; }

set

{

if (\_cardNumber != value)

{

\_cardNumber = value;

OnPropertyChanged(nameof(CardNumber));

}

}

}

public string Name

{

get { return \_name; }

set

{

if (\_name != value)

{

\_name = value;

OnPropertyChanged(nameof(Name));

}

}

}

public decimal Balance

{

get { return \_balance; }

set

{

if (\_balance != value)

{

\_balance = value;

OnPropertyChanged(nameof(Balance));

}

}

}

public Account(string cardNumber, string pin)

{

Database database = new Database();

CardNumber = cardNumber;

Pin = pin;

Name = database.GetName(CardNumber);

GmailAddress = database.GetGmail(CardNumber);

Balance = database.GetBalance(cardNumber, pin);

}

protected virtual void OnPropertyChanged(string propName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

}

}

}

**AutomatedTellerMachine.cs:**

using System;

using System.Collections.Generic;

namespace ATMClassLib

{

public class AutomatedTellerMachine

{

private decimal moneyAmount;

private string atmId;

private string address;

private string manufacturer;

private string model;

private string softwareVersion;

public string Address

{

get { return address; }

set { address = value; }

}

public decimal MoneyAmount

{

get { return moneyAmount; }

set { moneyAmount = value; }

}

public string Model

{

get { return model; }

set { model = value; }

}

public string SoftwareVersion

{

get { return softwareVersion; }

set { softwareVersion = value; }

}

public string Manufacturer

{

get { return manufacturer; }

set { manufacturer = value; }

}

public string ATMId

{

get { return atmId; }

set { atmId = value; }

}

public AutomatedTellerMachine(string atmId, decimal initialMoneyAmount, string address, string manufacturer, string model, string softwareVersion)

{

this.atmId = atmId;

this.moneyAmount = initialMoneyAmount;

this.address = address;

this.manufacturer = manufacturer;

this.model = model;

this.softwareVersion = softwareVersion;

}

}

}

**Bank.cs:**

using System;

using System.Collections.Generic;

using System.Net.Mail;

using System.Net;

namespace ATMClassLib

{

public delegate void SuccessfulOperationEventHandler(object sender, SuccessfulOperationEventArgs e);

public class SuccessfulOperationEventArgs : EventArgs

{

public string Parameter { get; }

public string Operator { get; }

public string Gmail { get; }

public SuccessfulOperationEventArgs(string parameter, string \_operator, string gmail)

{

Parameter = parameter;

Operator = \_operator;

Gmail = gmail;

}

}

public class Bank

{

private Database \_database;

public string Name { get; set; }

private List<AutomatedTellerMachine> atms;

public event SuccessfulOperationEventHandler SuccessfulOperation;

public List<AutomatedTellerMachine> ATMs

{

get { return atms; }

set { atms = value; }

}

public Bank(string bankName)

{

Name = bankName;

this.atms = new List<AutomatedTellerMachine>();

}

public Database GetDatabase()

{

return \_database;

}

public void SendMessage(string title, string subtitle, string gmail)

{

string sender = "ipz223\_pyuv@student.ztu.edu.ua";

string pass = "605845";

var smtpClient = new SmtpClient("smtp.gmail.com")

{

Port = 587,

UseDefaultCredentials = false,

Credentials = new NetworkCredential(sender, pass),

EnableSsl = true,

DeliveryMethod = SmtpDeliveryMethod.Network

};

smtpClient.Send(sender, gmail,title, subtitle);

OnSuccessfull(new SuccessfulOperationEventArgs(title, subtitle, gmail));

}

protected virtual void OnSuccessfull(SuccessfulOperationEventArgs e)

{

SuccessfulOperation?.Invoke(this, e);

}

}

}

**Database.cs:**

using System;

using System.Collections.Generic;

namespace ATMClassLib

{

public class Database

{

private readonly Dictionary<string, Account> accounts;

public delegate void TransactionDelegate(int accountId, decimal amount, string description);

public event TransactionDelegate TransactionEvent;

public Database()

{

this.accounts = new Dictionary<string, Account>();

accounts.Add("1234561234512345", new Account("1234561234512345", "James Conrod", "1234", 1000.0m, "jamescronrod5276@gmail.com"));

accounts.Add("9876549876598765", new Account("9876549876598765", "Nick Williams", "4321", 500.0m, "nickwilliams6498@gmail.com"));

accounts.Add("3333333333333333", new Account("3333333333333333", "Edward Elrick", "1111", 10279.120m, "edwardelrick1876@gmail.com"));

accounts.Add("4444444444444444", new Account("4444444444444444", "Denis Petrov", "2222", 8943.217m, "ipz223\_pyuv@gmail.com"));

accounts.Add("", new Account("", "", "", 0, " ipz223\_pyuv@gmail.com"));

}

public decimal GetBalance(string cardNumber, string pin)

{

if (!IsValidPin(cardNumber, pin))

{

return -1;

}

return accounts[cardNumber].Balance;

}

public bool IsCardValid(string cardNumber)

{

return accounts.ContainsKey(cardNumber);

}

public string GetCard(string cardNumber)

{

return accounts[cardNumber].CardImage;

}

public int UpdateBalance(string cardNumber, decimal amount)

{

accounts[cardNumber].Balance += amount;

OnTransactionEvent(accounts[cardNumber].Id, amount, "Переказ на карту");

accounts[cardNumber].AddTransaction(amount, "Переказ на карту");

return 1;

}

public int SendMoney(string senderCardNumber, string senderPin, string recipientCardNumber, decimal amount)

{

if (!IsValidPin(senderCardNumber, senderPin))

{

return 0;

}

if (!IsCardValid(recipientCardNumber))

{

return -1;

}

decimal senderBalance = GetBalance(senderCardNumber, senderPin);

if (senderBalance < amount)

{

return -2;

}

accounts[senderCardNumber].Balance -= amount;

accounts[recipientCardNumber].Balance += amount;

accounts[senderCardNumber].AddTransaction(-amount, $"Sent to {recipientCardNumber}");

accounts[recipientCardNumber].AddTransaction(amount, $"Received from {senderCardNumber}");

OnTransactionEvent(accounts[senderCardNumber].Id, -amount, $"Sent to {recipientCardNumber}");

OnTransactionEvent(accounts[recipientCardNumber].Id, amount, $"Received from {senderCardNumber}");

return 1;

}

public int AddTransaction(string cardNumber, decimal amount, string description)

{

if (accounts.ContainsKey(cardNumber))

{

OnTransactionEvent(accounts[cardNumber].Id, amount, description);

accounts[cardNumber].AddTransaction(amount, description);

return 1;

}

return 0;

}

public string GetName(string cardNumber)

{

return accounts[cardNumber].OwnerName;

}

public string GetGmail(string cardNumber)

{

return accounts[cardNumber].Gmail;

}

public string GetColor(string cardNumber)

{

return accounts[cardNumber].Color;

}

public int WithdrawMoney(string cardNumber, string pin, decimal amount)

{

if (!IsValidPin(cardNumber, pin))

{

return 0;

}

decimal currentBalance = GetBalance(cardNumber, pin);

if (currentBalance < amount)

{

return -1;

}

accounts[cardNumber].Balance -= amount;

OnTransactionEvent(accounts[cardNumber].Id, -amount, "Зняття коштів");

accounts[cardNumber].AddTransaction(-amount, "Зняття коштів");

return 1;

}

public bool IsValidPin(string cardNumber, string pin)

{

if (accounts.ContainsKey(cardNumber))

{

return accounts[cardNumber].Pin == pin.Trim();

}

return false;

}

private void OnTransactionEvent(int accountId, decimal amount, string description)

{

TransactionEvent?.Invoke(accountId, amount, description);

}

private class Account

{

private static int nextId = 1;

public int Id { get; }

public string CardNumber { get; }

public string Gmail { get; }

public string CardImage { get; }

public string Color { get; }

public string OwnerName { get; }

public string Pin { get; }

public decimal Balance { get; set; }

public List<Transaction> Transactions { get; }

public Account(string cardNumber, string ownerName, string pin, decimal balance, string gmail)

{

Id = nextId++;

CardNumber = cardNumber;

Gmail = gmail;

OwnerName = ownerName;

Pin = pin;

Balance = balance;

Transactions = new List<Transaction>();

Gmail = gmail;

}

public void AddTransaction(decimal amount, string description)

{

Transaction newTransaction = new Transaction(amount, description);

Transactions.Add(newTransaction);

}

}

private class Transaction

{

public decimal Amount { get; }

public string Description { get; }

public DateTime Timestamp { get; }

public Transaction(decimal amount, string description)

{

Amount = amount;

Description = description;

Timestamp = DateTime.Now;

}

}

}

}

**Результат виконання програми:**

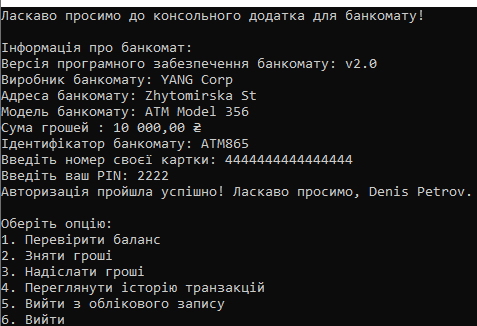


Рис. 1.1 Авторизація разом з головним меню.

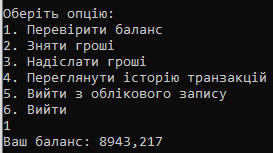


Рис. 1.2 Перевірка балансу.

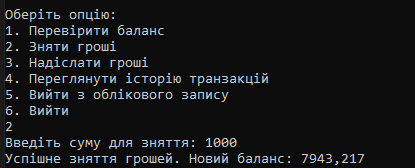


Рис. 1.3 Зняття грошей.

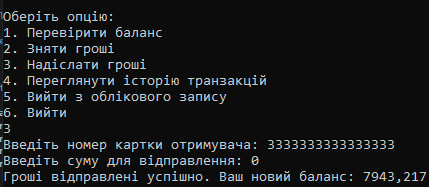


Рис. 1.4 Надсилання грошей.



Рис. 1.5 Історія транзакцій

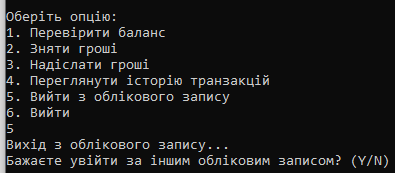


Рис. 1.6 Вихід із облікового запису.

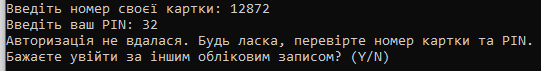


Рис. 1.7 Виведення помилки про некоректне введення даних.

**WPF:**

**HomeView.xaml:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ATMWPFApp.View

{

/// <summary>

/// Логика взаимодействия для HomeView.xaml

/// </summary>

public partial class HomeView : UserControl

{

public HomeView()

{

InitializeComponent();

}

private void CloseWithdraw(object sender, RoutedEventArgs e)

{

WithdrawBorder.Visibility = Visibility.Collapsed;

}

private void WithdrawMyCount\_Click(object sender, RoutedEventArgs e)

{

WithdrawMyCount.Visibility = Visibility.Collapsed;

}

private void OpenTopUp(object sender, RoutedEventArgs e)

{

TopUpBorder.Visibility = Visibility.Visible;

}

private void CloseTopUp(object sender, RoutedEventArgs e)

{

TopUpBorder.Visibility = Visibility.Collapsed;

}

private void OpenWithdraw(object sender, RoutedEventArgs e)

{

WithdrawBorder.Visibility = Visibility.Visible;

}

private void OpenTransferBorder(object sender, RoutedEventArgs e)

{

TransferBorder.Visibility = Visibility.Visible;

}

private void CloseTransferBorder(object sender, RoutedEventArgs e)

{

TransferBorder.Visibility = Visibility.Collapsed;

}

}

}

**AuthorisationViewModel.cs:**

using ATMClassLib;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Security.Principal;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

namespace ATMWPFApp.ViewModel

{

public class AuthorisationViewModel : INotifyPropertyChanged

{

private RelayCommand \_auth;

private MainViewModel \_mainViewModel;

private string \_inputText = string.Empty;

private string \_inputPin = string.Empty;

private HomeViewModel home;

public ICommand AuthCommand

{

get

{

if (\_auth == null)

{

\_auth = new RelayCommand(Authorisation);

}

return \_auth;

}

}

public AuthorisationViewModel()

{

home = new HomeViewModel(this);

}

public AuthorisationViewModel(MainViewModel mainViewModel)

{

\_mainViewModel = mainViewModel;

}

private void Authorisation(object parameter)

{

\_mainViewModel.SelectedVM = new HomeViewModel((AuthorisationViewModel)\_mainViewModel.SelectedVM);

}

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string propName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

}

public string InputText

{

get { return \_inputText; }

set

{

if (\_inputText != value)

{

\_inputText = value;

OnPropertyChanged(nameof(InputText));

}

}

}

public string InputPin

{

get { return \_inputPin; }

set

{

if (\_inputPin != value)

{

\_inputPin = value;

OnPropertyChanged(nameof(InputPin));

}

}

}

}

}

**HomeViewModel.cs:**

using ATMClassLib;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

using System.Windows;

namespace ATMWPFApp.ViewModel

{

public class HomeViewModel : INotifyPropertyChanged

{

private Account \_account;

AutomatedTellerMachine \_atm;

private Bank \_Bank;

private MainViewModel \_MainViewModel;

private RelayCommand \_withdrawBalance;

private RelayCommand \_topupBalance;

private RelayCommand \_transferBalance;

private string \_inputText = string.Empty;

private string \_inputAmount = string.Empty;

public string InputCard

{

get { return \_inputText; }

set

{

if (\_inputText != value)

{

\_inputText = value;

OnPropertyChanged(nameof(InputCard));

}

}

}

public string InputAmount

{

get { return \_inputAmount; }

set

{

if (\_inputAmount != value)

{

\_inputAmount = value;

OnPropertyChanged(nameof(InputCard));

}

}

}

public ICommand WithdrawBalance

{

get

{

if (\_withdrawBalance == null)

{

\_withdrawBalance = new RelayCommand(WithdrawMoney);

}

return \_withdrawBalance;

}

}

public ICommand TransferBalance

{

get

{

if (\_transferBalance == null)

{

\_transferBalance = new RelayCommand(SendMoney);

}

return \_transferBalance;

}

}

public ICommand TopUpBalance

{

get

{

if (\_topupBalance == null)

{

\_topupBalance = new RelayCommand(TopUpMoney);

}

return \_topupBalance;

}

}

private void TopUpMoney(object parameter)

{

Account.Balance += Convert.ToDecimal(parameter);

ATM.MoneyAmount += Convert.ToDecimal(parameter);

Bank bank = new Bank("Privat24");

bank.SendMessage(parameter.ToString(), "+", Account.GmailAddress);

MessageBox.Show($"Ви поповнили рахунок на {parameter} гривень");

bank.SuccessfulOperation += SuccessfulOperationHandler;

}

private static void SuccessfulOperationHandler(object sender, SuccessfulOperationEventArgs e)

{

MessageBox.Show($"Операція успішна: {e.Operator}{e.Parameter}.\nВідправлено на почту: {e.Gmail}");

}

private void WithdrawMoney(object parameter)

{

if (Account != null)

{

try

{

decimal balance = Account.Balance;

decimal Amount = Convert.ToDecimal(parameter);

if (balance >= Amount && ATM.MoneyAmount >= Amount)

{

Account.Balance -= Convert.ToDecimal(parameter);

Bank bank = new Bank("Privat24");

bank.SendMessage(parameter.ToString(), "-", Account.GmailAddress);

MessageBox.Show($"Ви зняли {parameter} гривень");

bank.SuccessfulOperation += SuccessfulOperationHandler;

}

else if (ATM.MoneyAmount < Amount)

{

MessageBox.Show("В терміналі недостатньо коштів.\nОперація не виконана");

}

else if (Amount > balance)

{

MessageBox.Show("Недостатньо коштів.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при знятті грошей: {ex.Message}");

}

}

else

{

MessageBox.Show("Account is not initialized.");

}

}

private void SendMoney(object parameter)

{

if (Account != null)

{

try

{

decimal balance = Account.Balance;

decimal Amount = Convert.ToDecimal(InputAmount);

Database database = new Database();

if (balance >= Amount && database.IsCardValid(InputCard) && InputCard != Account.CardNumber)

{

Account Account2 = new Account(InputCard, "");

Account.Balance -= Amount;

database.UpdateBalance(InputCard, Amount);

Bank bank = new Bank("Privat24");

bank.SendMessage(InputAmount, "Переказ на карту\n-", Account.GmailAddress);

bank.SendMessage(InputAmount, "Поповнення карти\n+", Account2.GmailAddress);

bank.SuccessfulOperation += SuccessfulOperationHandler;

MessageBox.Show($"Ви переказали {InputAmount} гривень\nНа карту {Account2.Name}");

}

else if (!database.IsCardValid(InputCard))

{

MessageBox.Show("🚫 Неправильний номер карти");

}

else if (InputCard == Account.CardNumber)

{

MessageBox.Show("🚫 Це ж ваша картка!");

}

else if (Amount > balance)

{

MessageBox.Show("Недостатньо коштів.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при знятті грошей: {ex.Message}");

}

}

else

{

MessageBox.Show("Account is not initialized.");

}

}

public HomeViewModel()

{

\_MainViewModel = new MainViewModel();

if (\_account == null)

{

\_atm = new AutomatedTellerMachine("ATM865", 10000, "Київська 68", "YANG Corp", "ATM Model 356", "v2.0");

\_account = new Account("1111111111111111", "1111");

}

}

public HomeViewModel(AuthorisationViewModel authViewModel)

{

Database database = new Database();

if (database.IsCardValid(authViewModel.InputText))

{

if (!database.IsValidPin(authViewModel.InputText, authViewModel.InputPin))

{

MessageBox.Show("🚫 Неправильний пін карти");

\_MainViewModel.SelectedVM = new AuthorisationViewModel();

}

else

{

\_Bank = new Bank("Tantanbank");

\_account = new Account(authViewModel.InputText, authViewModel.InputPin);

\_atm = new AutomatedTellerMachine("ATM865", 10000, "Київська 68", "YANG Corp", "ATM Model 356", "v2.0");

}

}

else

{

MessageBox.Show("🚫 Неправильний номер карти");

}

}

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string propName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

}

public Account Account

{

get { return \_account; }

set

{

if (\_account != value)

{

\_account = value;

OnPropertyChanged(nameof(Account));

}

}

}

public AutomatedTellerMachine ATM

{

get { return \_atm; }

set

{

if (\_atm != value)

{

\_atm = value;

OnPropertyChanged(nameof(ATM));

}

}

}

public Bank Bank

{

get { return \_Bank; }

set

{

if (\_Bank != value)

{

\_Bank = value;

OnPropertyChanged(nameof(Bank));

}

}

}

}

}

**MainViewModel.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ATMWPFApp.ViewModel

{

public class MainViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string propName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

}

private object \_selectedVM;

public object SelectedVM

{

get => \_selectedVM;

set { \_selectedVM = value; OnPropertyChanged("SelectedVM"); }

}

public MainViewModel()

{

SelectedVM = new AuthorisationViewModel(this);

}

}

}

**Результат виконання програми:**

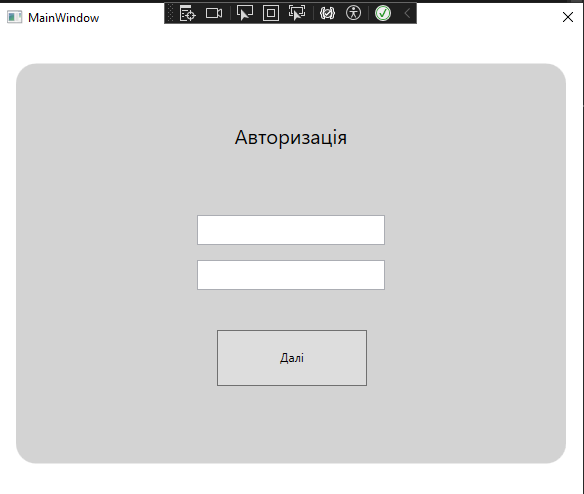


Рис 2.1 Авторизація.

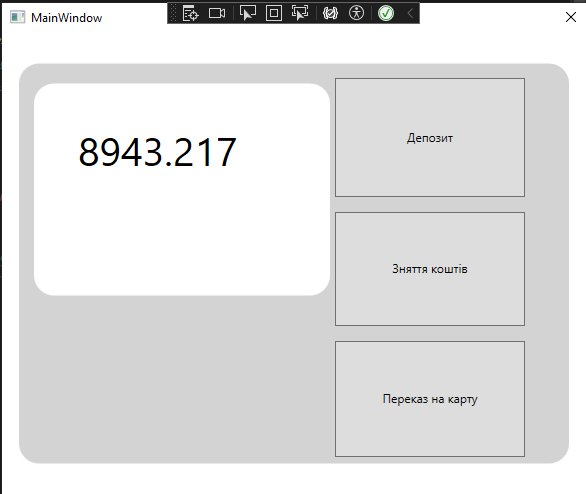
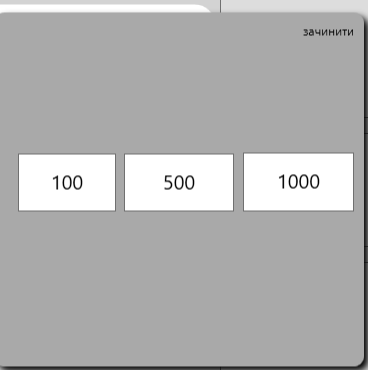
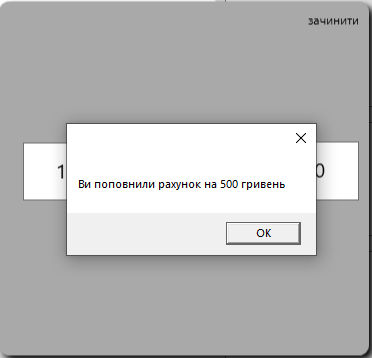


Рис. 2.2 Візуальний вигляд.





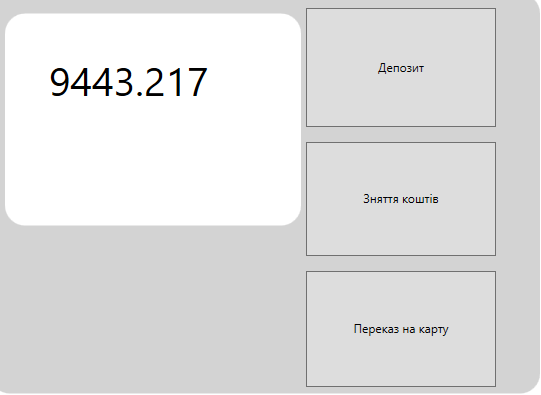
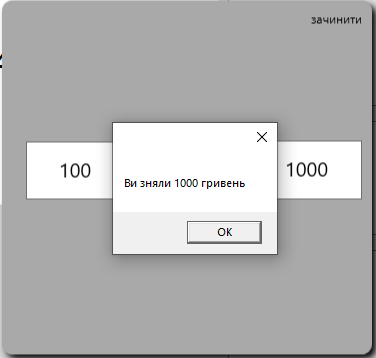


Рис. 2.3 Депозит.



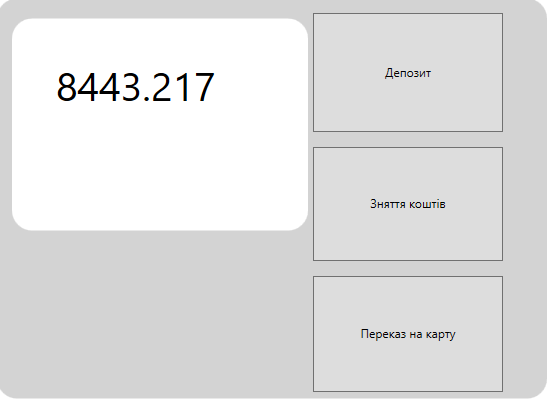
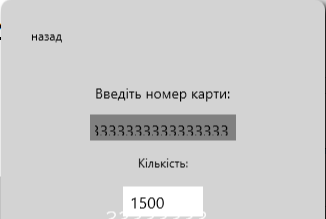


Рис. 2.4 Зняття коштів з карти.



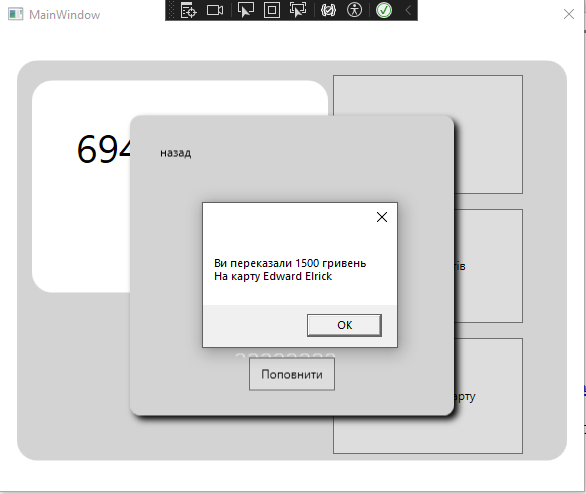


Рис 2.5 Переказ коштів за картою.

**Репозиторій GitLab:** <https://gitlab.com/2022-2026/ipz-22-4/denis-volodymyrovych-petrov/dotnetlab_01_petrov>

**Висновок:** навчився використовувати оголошувати та використовувати делегати та події у мові програмування C#.