**Лабораторна робота No 1**  
Тема: Використання делегатів та подій у C#.

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

**Хiд рoбoти:**

Завдання 1. 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 проекті.

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.

Передбачити можливості:

● аутентифікація (перевірка введення номеру картки та пін-коду);

● перегляд балансу на картці;

● зняття коштів;

● зарахування коштів на картку;

● перерахування коштів на картку із заданим номером.

Повинна бути передбачена можливість додавання власних обробників подій на

виконання операцій аутентифікації, перегляду балансу, зняття коштів та перерахування

коштів на іншу картку.

Додайте власні обробники подій, які будуть виводити повідомлення:

● для віконного додатку - за допомогою виклику

MessageBox.Show("Повідомлення");

● для консольного додатку - за допомогою виклику

Console.WriteLine("Повідомлення");

Бібліотека класів:

*Лiстинг:*

*namespace* BankSimulation

{

public class Account

{

public int CardNumber { get; }

public string OwnerName { get; }

public decimal Balance { get; private set; }

public decimal Pin { get; }

public Account(int cardNumber, string ownerName, decimal initialBalance, decimal pin)

{

CardNumber = cardNumber;

OwnerName = ownerName;

Balance = initialBalance;

Pin = pin;

}

public event Action<decimal> BalanceChanged;

public void Withdraw(decimal amount)

{

if (Balance >= amount)

{

Balance -= amount;

BalanceChanged(Balance);

}

}

public void Deposit(decimal amount)

{

Balance += amount;

BalanceChanged(Balance);

}

public bool Authenticate(decimal pin)

{

return Pin == pin;

}

}

public class AutomatedTellerMachine

{

public string ATMId { get; }

public decimal AvailableFunds { get; private set; }

public event Action<string, decimal> AvailableFundsChanged;

public AutomatedTellerMachine(string atmId, decimal initialFunds)

{

ATMId = atmId;

AvailableFunds = initialFunds;

}

public void Withdraw(decimal amount)

{

if (amount <= AvailableFunds)

{

AvailableFunds -= amount;

AvailableFundsChanged(ATMId, AvailableFunds);

}

}

public void Deposit(decimal amount)

{

AvailableFunds += amount;

AvailableFundsChanged(ATMId, AvailableFunds);

}

}

public class Bank

{

public string BankName { get; }

public List<AutomatedTellerMachine> ATMs { get; } = new List<AutomatedTellerMachine>();

public List<Account> Accounts { get; } = new List<Account>();

private AutomatedTellerMachine fixedATM; // Фіксований банкомат для клієнта

public event Action<string> LogEvent;

public Bank(string bankName)

{

BankName = bankName;

}

public void AddATM(AutomatedTellerMachine atm)

{

ATMs.Add(atm);

atm.AvailableFundsChanged += HandleATMAvailableFundsChanged;

}

public void CreateAccount(int cardNumber, string ownerName, decimal initialBalance, decimal pin)

{

var account = new Account(cardNumber, ownerName, initialBalance, pin);

Accounts.Add(account);

account.BalanceChanged += newBalance => LogEvent($"Зміна балансу для {account.OwnerName} (Номер картки: {account.CardNumber}): {newBalance} грн.");

if (LogEvent != null)

{

LogEvent($"Створено новий обліковий запис для {ownerName} (Номер картки: {cardNumber}).");

}

}

public Account Authenticate(int cardNumber, decimal pin)

{

var account = Accounts.FirstOrDefault(acc => acc.CardNumber == cardNumber && acc.Pin == pin);

if (account == null)

{

LogEvent?.Invoke($"Невірний номер картки або пін-код для картки {cardNumber}.");

}

return account;

}

public void ShowAccountBalance(Account account)

{

LogEvent($"Баланс для {account.OwnerName} (Номер картки: {account.CardNumber}): {account.Balance} грн.");

}

public void SetFixedATM(AutomatedTellerMachine atm)

{

fixedATM = atm;

}

public void WithdrawFunds(Account account, decimal amount)

{

if (amount > 0)

{

if (fixedATM != null)

{

if (fixedATM.AvailableFunds >= amount)

{

if (account.Balance >= amount)

{

account.Withdraw(amount);

fixedATM.Withdraw(amount);

LogEvent($"Знято {amount} грн з рахунку {account.OwnerName}. Новий баланс: {account.Balance} грн.");

}

else

{

LogEvent("Недостатньо коштів на рахунку.");

}

}

else

{

LogEvent($"Недостатньо коштів у банкоматі {fixedATM.ATMId}.");

}

}

else

{

LogEvent("Банкомат не вибраний.");

}

}

else

{

LogEvent("Сума для зняття повинна бути більше нуля.");

}

}

public void DepositFunds(Account account, decimal amount)

{

if (amount > 0)

{

if (fixedATM != null)

{

account.Deposit(amount);

fixedATM.Deposit(amount);

LogEvent($"Поповнено {amount} грн на рахунок {account.OwnerName}. Новий баланс: {account.Balance} грн.");

}

else

{

LogEvent("Банкомат не вибраний.");

}

}

else

{

LogEvent("Сума для поповнення повинна бути більше нуля.");

}

}

public void TransferFunds(Account sourceAccount, Account targetAccount, decimal amount)

{

if (sourceAccount.Balance < amount)

{

LogEvent("Недостатньо коштів для переказу.");

}

else

{

sourceAccount.Withdraw(amount);

targetAccount.Deposit(amount);

LogEvent($"Переказано {amount} грн з рахунку {sourceAccount.OwnerName} на рахунок {targetAccount.OwnerName}.");

}

}

public void HandleATMAvailableFundsChanged(string atmId, decimal newFunds)

{

LogEvent?.Invoke($"Зміна доступних коштів в банкоматі {atmId}. Новий баланс: {newFunds} грн.");

}

}

}

Консольний проект:

*Лiстинг прoгрaми:*

using System;

using System.Collections.Generic;

using System.Linq;

using BankSimulation;

namespace BankSimulation

{

class Program

{

static void Main(string[] args)

{

var bank = new Bank("MyBank");

// Створюємо облікові записи

bank.CreateAccount(1234, "John Doe", 11000, 5678);

bank.CreateAccount(5678, "Jane Smith", 1500, 4321);

// Створюємо банкомати та додаємо обробники подій

var atm1 = new AutomatedTellerMachine("ATM1", 10000);

var atm2 = new AutomatedTellerMachine("ATM2", 15000);

bank.AddATM(atm1);

bank.AddATM(atm2);

bank.SetFixedATM(atm1);

bank.LogEvent += LogToConsole;

while (true)

{

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

if (int.TryParse(Console.ReadLine(), out int cardNumber))

{

Console.WriteLine("Введіть пін-код:");

if (decimal.TryParse(Console.ReadLine(), out decimal pin))

{

Account account = bank.Authenticate(cardNumber, pin);

if (account != null)

{

LogToConsole($"Ласкаво просимо, {account.OwnerName}!");

ShowMenu(bank, account);

}

}

else

{

LogToConsole("Невірний формат пін-коду.");

}

}

else

{

LogToConsole("Невірний формат номеру картки.");

}

}

}

static void ShowMenu(Bank bank, Account account)

{

while (true)

{

Console.WriteLine("Оберіть опцію:");

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

Console.WriteLine("2. Зняти кошти");

Console.WriteLine("3. Поповнити рахунок");

Console.WriteLine("4. Переказати кошти");

Console.WriteLine("5. Вийти");

int choice;

if (int.TryParse(Console.ReadLine(), out choice))

{

switch (choice)

{

case 1:

bank.ShowAccountBalance(account);

break;

case 2: // Зняти кошти

Console.WriteLine("Введіть суму для зняття:");

if (decimal.TryParse(Console.ReadLine(), out decimal withdrawalAmount))

{

bank.WithdrawFunds(account, withdrawalAmount);

}

else

{

Console.WriteLine("Невірний формат суми.");

}

break;

case 3: // Поповнити рахунок

Console.WriteLine("Введіть суму для поповнення:");

if (decimal.TryParse(Console.ReadLine(), out decimal depositAmount))

{

bank.DepositFunds(account, depositAmount);

}

else

{

Console.WriteLine("Невірний формат суми.");

}

break;

case 4:

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

if (int.TryParse(Console.ReadLine(), out int targetCardNumber))

{

var targetAccount = bank.Accounts.FirstOrDefault(acc => acc.CardNumber == targetCardNumber);

if (targetAccount != null)

{

Console.WriteLine("Введіть суму для переказу:");

if (decimal.TryParse(Console.ReadLine(), out decimal transferAmount))

{

bank.TransferFunds(account, targetAccount, transferAmount);

}

else

{

Console.WriteLine("Невірний формат суми.");

}

}

else

{

LogToConsole("Обліковий запис отримувача не знайдено.");

}

}

else

{

LogToConsole("Невірний формат номеру картки.");

}

break;

case 5:

LogToConsole("Вихід з облікового запису.");

return;

default:

LogToConsole("Невірний вибір.");

break;

}

}

else

{

LogToConsole("Невірний вибір.");

}

}

}

static void LogToConsole(string message)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine(message);

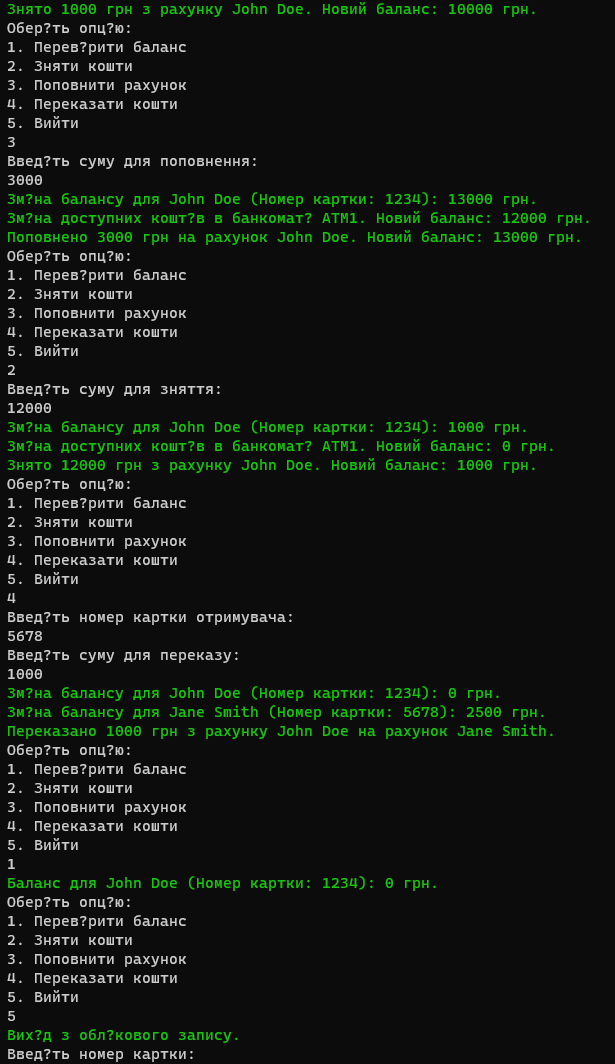
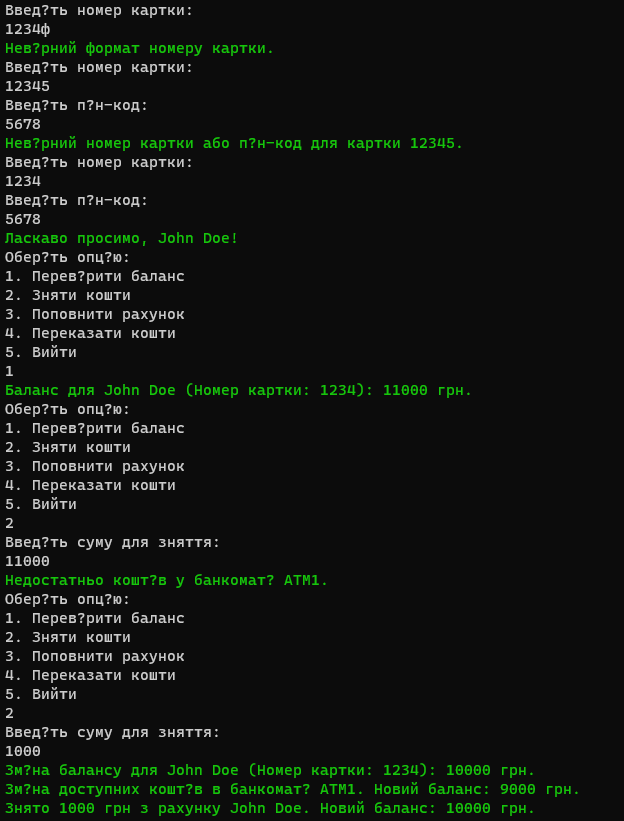
Console.ResetColor();

}

}

}

*Результaти:*



**Віконний проект:**

*Лістинг програми:*

*using* static System.Windows.Forms.VisualStyles.VisualStyleElement.Rebar;

using System.Security.Principal;

using BankSimulation;

namespace WinFormsApp1

{

public partial class Form1 : Form

{

UserControl1 UserControlAuthorization = new UserControl1();

UserControlMenu UserControlMenu = new UserControlMenu();

UserControlDeposit UserControlDeposit = new UserControlDeposit();

UserControlWithdraw UserControlWithdraw = new UserControlWithdraw();

UserControlTransfer UserControlTransfer = new UserControlTransfer();

UserControlCheckBalance UserControlCheckBalance = new UserControlCheckBalance();

Bank bank = new Bank("MyBank");

private Account account;

public Form1()

{

InitializeComponent();

this.Controls.Add(UserControlAuthorization);

bank.CreateAccount(123456, "User1", 1100, 1111);

bank.CreateAccount(789012, "User2", 12000, 2222);

var atm1 = new AutomatedTellerMachine("ATM1", 10000);

var atm2 = new AutomatedTellerMachine("ATM2", 15000);

bank.AddATM(atm1);

bank.AddATM(atm2);

bank.SetFixedATM(atm1);

bank.LogEvent += LogToConsole;

// Додавання облікових записів

// Обробник події натискання кнопки "Підтвердити" на формі авторизації

// Обробник події натискання кнопки "Підтвердити" на формі авторизації

UserControlAuthorization.buttonAprove.Click += (s, e) =>

{

if (int.TryParse(UserControlAuthorization.textBoxCardNum.Text, out int cardNumber))

{

if (decimal.TryParse(UserControlAuthorization.textBoxPass.Text, out decimal pin))

{

// Автентифікація

account = bank.Authenticate(cardNumber, pin);

if (account != null)

{

// Якщо автентифікація успішна, перехід до меню

this.Controls.Clear();

this.Controls.Add(UserControlMenu);

}

else

{

LogToConsole("Невірний номер картки або пін-код.");

}

}

else

{

LogToConsole("Невірний формат пін-коду.");

}

}

else

{

LogToConsole("Невірний формат номеру картки.");

}

};

// Обробники подій для кнопок меню

UserControlMenu.buttonDeposit.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlDeposit);

};

UserControlMenu.buttonWithDraw.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlWithdraw);

};

UserControlMenu.buttonTransfer.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlTransfer);

};

UserControlMenu.buttonCheckBalance.Click += (s, e) =>

{

// Виведення номера картки з заміною всіх, крім останніх 4 цифр, на зірочки

string cardNumber = account.CardNumber.ToString();

string cardNumberStars = cardNumber.Length > 4

? new string('\*', cardNumber.Length - 4) + cardNumber.Substring(cardNumber.Length - 4)

: cardNumber;

UserControlCheckBalance.labelCardNumber.Text = cardNumberStars;

this.Controls.Clear();

this.Controls.Add(UserControlCheckBalance);

};

// Обробник події для кнопки "Назад" в меню

UserControlMenu.buttonBack.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlAuthorization);

};

// Обробники подій для кнопок "Назад"

UserControlDeposit.buttonBack.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlMenu);

};

UserControlWithdraw.buttonBack.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlMenu);

};

UserControlTransfer.buttonBack.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlMenu);

};

UserControlCheckBalance.buttonBack.Click += (s, e) =>

{

this.Controls.Clear();

this.Controls.Add(UserControlMenu);

};

// Обробники подій для кнопок "Вийти"

UserControlAuthorization.buttonExit.Click += (s, e) => { Application.Exit(); };

UserControlMenu.buttonExit.Click += (s, e) => { Application.Exit(); };

UserControlDeposit.buttonExit.Click += (s, e) => { Application.Exit(); };

UserControlWithdraw.buttonExit.Click += (s, e) => { Application.Exit(); };

UserControlTransfer.buttonExit.Click += (s, e) => { Application.Exit(); };

UserControlCheckBalance.buttonExit.Click += (s, e) => { Application.Exit(); };

// Обробники подій для кнопок "Поповнити", "Зняти", "Переказати"

UserControlDeposit.buttonDeposit.Click += (s, e) =>

{

if (decimal.TryParse(UserControlDeposit.textBoxCardNum.Text, out decimal depositAmount))

{

bank.DepositFunds(account, depositAmount);

}

else

{

LogToConsole("Невірний формат суми.");

}

};

UserControlWithdraw.buttonWithDraw.Click += (s, e) =>

{

if (decimal.TryParse(UserControlWithdraw.textBoxSumWith.Text, out decimal withdrawalAmount))

{

bank.WithdrawFunds(account, withdrawalAmount);

}

else

{

LogToConsole("Невірний формат суми.");

}

};

UserControlTransfer.buttonTransfer.Click += (s, e) =>

{

if (int.TryParse(UserControlTransfer.textBoxCardNum.Text, out int targetCardNumber))

{

Account targetAccount = bank.Accounts.FirstOrDefault(acc => acc.CardNumber == targetCardNumber);

if (targetAccount != null)

{

if (decimal.TryParse(UserControlTransfer.textBoxSumWith.Text, out decimal transferAmount))

{

bank.TransferFunds(account, targetAccount, transferAmount);

}

else

{

LogToConsole("Невірний формат суми.");

}

}

else

{

LogToConsole("Обліковий запис отримувача не знайдено.");

}

}

else

{

LogToConsole("Невірний формат номеру картки.");

}

};

// Обробник події для кнопки "Перевірити баланс"

UserControlCheckBalance.buttonCheckBalance.Click += (s, e) =>

{

bank.ShowAccountBalance(account);

};

static void LogToConsole(string message)

{

MessageBox.Show(message);

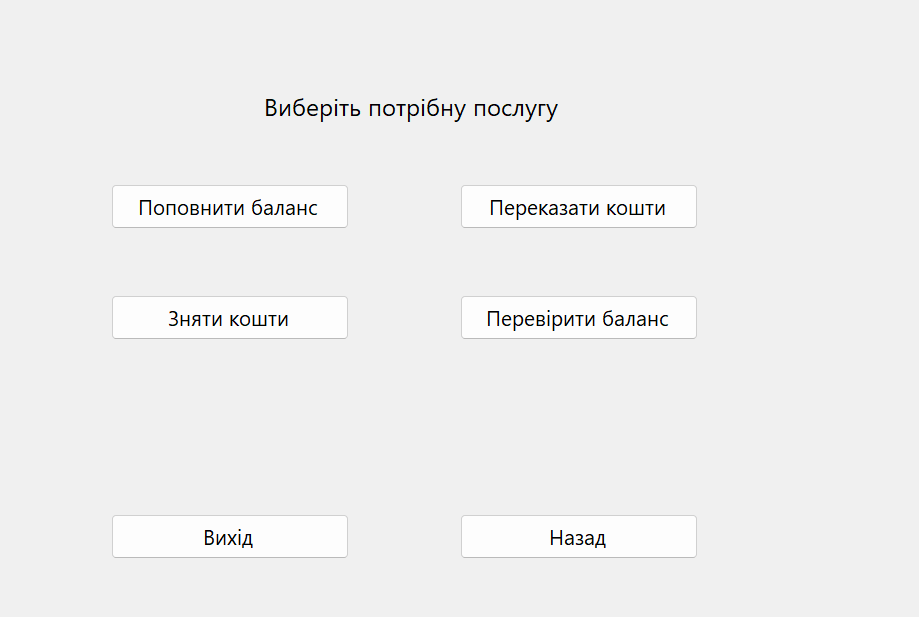
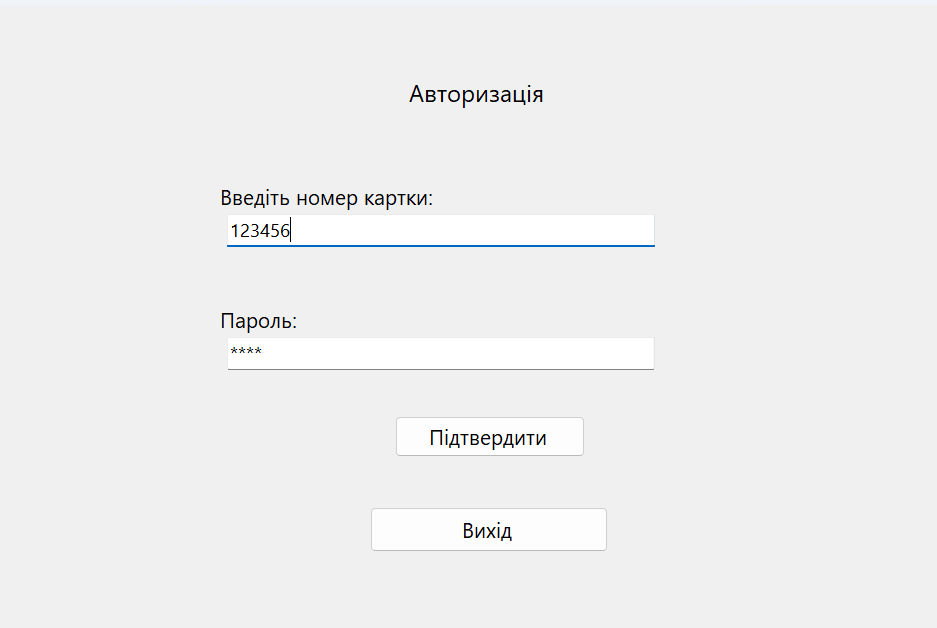
}

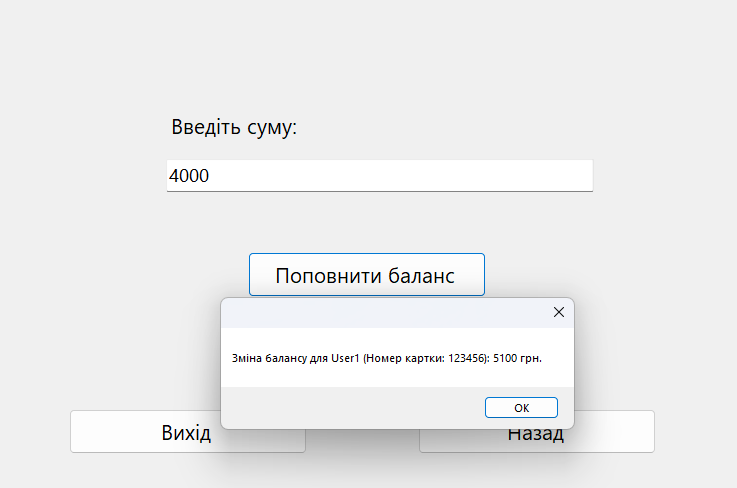
}

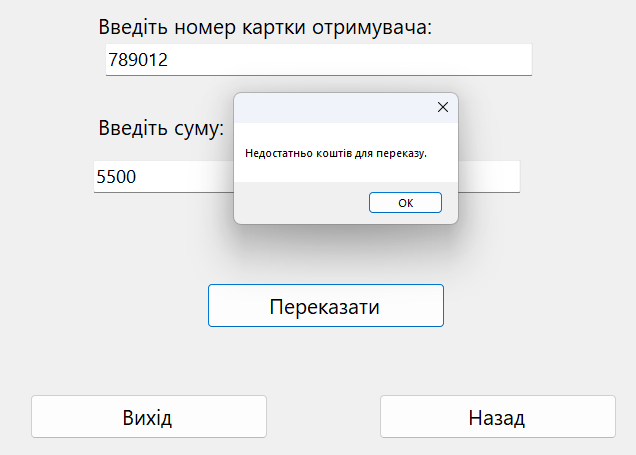
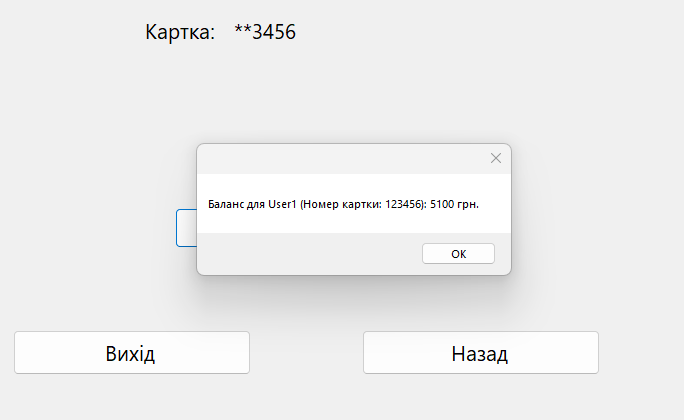
}

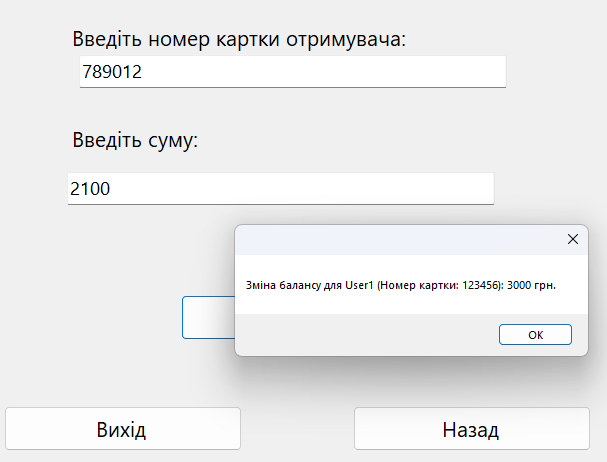
}

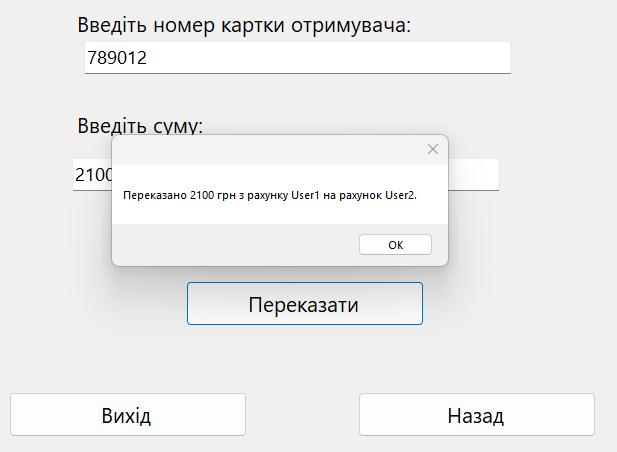
*Результати:*

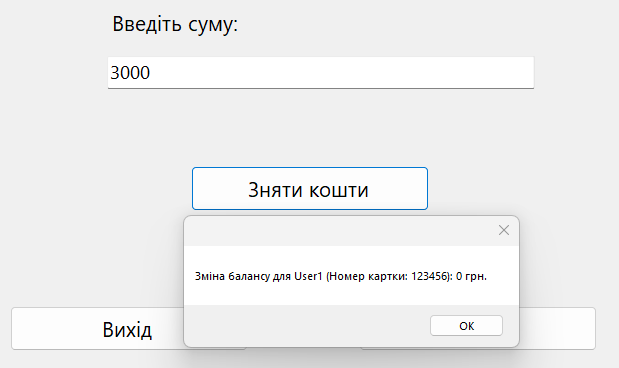
**

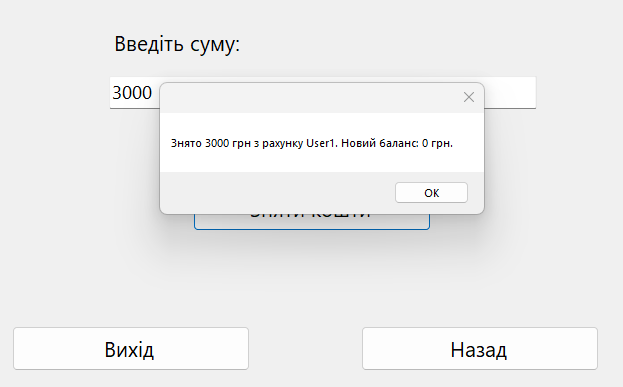


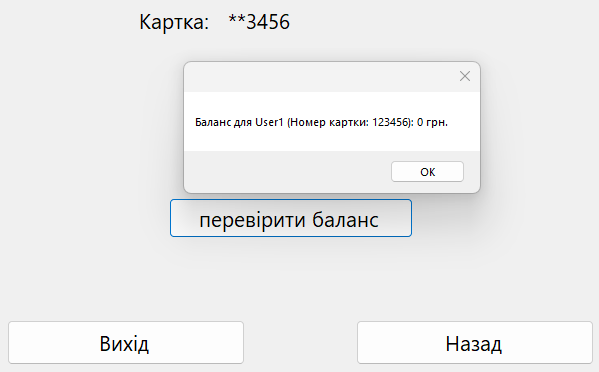












**Висновок:** у цій лабораторній я навчився оголошувати та використовувати делегати та події у мові програмування C#.