

## Лабораторна робота №1

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

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

**Хід роботи:**

### 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");`

Лістинг класу “Account”:

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

namespace ClassLibraryBank
{
```

```

public delegate void AccountHandler(string message);

public class Account
{
    public delegate void BankOperationsHandler(Account sender,
OperationsBankEventArgs e);

    public static int amountUsers = 0;
    public string Name { get; set; }
    public string Surname { get; set; }
    public string CardNumber { get; private set; }
    public float Balance { get; set; }
    public int Pin { get; private set; }
    private float balanceUsd;
    private float exchangeRate;

    AccountHandler info;

    public Account(string name, string surname, string cardNumber, int pin, float
balance)
    {
        Name = name;
        Surname = surname;
        CardNumber = cardNumber;
        Pin = pin;
        Balance = balance;
        amountUsers++;
        this.balanceUsd = 0;
        this.exchangeRate = 38;
    }
    public float BalanceUsd
    {
        get { return balanceUsd; }
        set { balanceUsd = value; }
    }

    public float ExchangeRate
    {
        get { return exchangeRate; }
        set { exchangeRate = value; }
    }
    public void RegisterHandler(AccountHandler del)
    {
        info = del;
    }
    public void AddUsd(float amount)
    {
        balanceUsd += amount;
    }

    public void SubtractUsd(float amount)
    {
        balanceUsd -= amount;
    }

    public void PrintInfo()
    {
        info?.Invoke($"Ім'я: {Name}\nПрізвище: {Surname}\nНомер картки:
{CardNumber}\nБаланс: {Balance}");
    }

    public void PrintBalance()
    {

```

```

        info?.Invoke($"Баланс: {Balance}");
    }
}

```

Лістинг класу “OperationsBankEventArgs”:

```

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

namespace ClassLibraryBank
{
    public class OperationsBankEventArgs
    {
        public string Message { get; }
        public float Sum { get; }
        public OperationsBankEventArgs(string message, float sum)
        {
            Message = message;
            Sum = sum;
        }
        public float EquivalentUAN { get; set; }
        public float BalanceUsd { get; set; }
    }
}

```

Лістинг класу “AutomatedMachine ”:

```

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

namespace ClassLibraryBank
{
    public class AutomatedTellerMachine
    {
        private float ConvertUsdToUan(float usdAmount, float exchangeRate)
        {
            return usdAmount * exchangeRate;
        }
        public delegate void BankOperationsHandler(AutomatedTellerMachine sender,
OperationsBankEventArgs e);
        public event BankOperationsHandler Notify;
        public float Money { get; set; }
        public string IdATMs { get; protected set; }
        public string Address { get; protected set; }

        public AutomatedTellerMachine(float money, string idATMs, string address)
        {
            Money = money;
            IdATMs = idATMs;
            Address = address;
        }

        public void Put(float sum, Account[] account, int user)
        {
            account[user].Balance += sum;
            Money += sum;
        }
    }
}

```

```

        Notify?.Invoke(this, new OperationsBankEventArgs($"На рахунок зараховано
{sum} UAN", sum));
    }

    public void Take(float sum, Account[] account, int user)
    {
        if (account[user] != null)
        {
            if (Money >= sum)
            {
                if (account[user].Balance >= sum)
                {
                    account[user].Balance -= sum;
                    Money -= sum;
                    Notify?.Invoke(this, new OperationsBankEventArgs($"{sum} UAN було
знято з рахунку", sum));
                }
                else
                {
                    Notify?.Invoke(this, new OperationsBankEventArgs("Недостатньо
коштів на рахунку", sum));
                }
            }
            else
            {
                Notify?.Invoke(this, new OperationsBankEventArgs("Вибачте, наразі ви
не можете зняти кошти через технічні проблеми банку", sum));
            }
        }
    }

    public void TakeUsd(float sum, Account[] account, int user)
    {
        float equivalentUan = sum * account[user].ExchangeRate;

        if (account[user] != null)
        {
            if (Money >= equivalentUan)
            {
                if (account[user].BalanceUsd >= sum)
                {
                    account[user].SubtractUsd(sum);
                    Money -= equivalentUan;
                    Notify?.Invoke(this, new OperationsBankEventArgs($"{sum}
{currentCurrency} було знято з рахунку", equivalentUan));
                }
                else
                {
                    Notify?.Invoke(this, new OperationsBankEventArgs($"Недостатньо
коштів на рахунку в {currentCurrency}", equivalentUan));
                }
            }
            else
            {
                Notify?.Invoke(this, new OperationsBankEventArgs($"Вибачте, наразі ви
не можете зняти кошти в {currentCurrency} через технічні проблеми банку",
equivalentUan));
            }
        }
    }

    public void PutUsd(float sum, Account[] accounts, int user)
    {
        float equivalentUan = ConvertUsdToUan(sum, accounts[user].ExchangeRate);
    }

```

```

        accounts[user].BalanceUsd += sum;
        accounts[user].Balance += equivalentUan;

        Notify?.Invoke(this, new OperationsBankEventArgs($"Поповнено рахунок на {sum}
USD ({equivalentUan} UAN) в доларах.", equivalentUan));
    }

    public bool CardTransfer(float sum, string cardNumber, Account[] account, int
user)
    {
        bool check = false;
        if (account[user].Balance >= sum)
        {
            if (!check)
            {
                for (int j = 0; j < Account.amountUsers; j++)
                {
                    if (account[j].CardNumber == cardNumber)
                    {
                        check = true;
                        float equivalentSum = sum * account[user].ExchangeRate;
                        account[j].Balance += equivalentSum;
                        account[user].Balance -= sum;
                        Notify?.Invoke(this, new
OperationsBankEventArgs($" {equivalentSum} UAN було переведено на картку за номером
{cardNumber}", equivalentSum));
                        return true;
                    }
                }

                if (!check)
                {
                    Notify?.Invoke(this, new OperationsBankEventArgs($"Картку за но-
мером {cardNumber} не знайдено", sum));
                }
            }
            else
            {
                Notify?.Invoke(this, new OperationsBankEventArgs("Недостатньо коштів на
рахунку", sum));
                return false;
            }
            return false;
        }
        private Currency currentCurrency = Currency.UAH;

        public enum Currency
        {
            UAH,
            USD
        }

        public void SetCurrency(Currency currency)
        {
            currentCurrency = currency;
        }
        public Currency GetCurrency()
        {
            return currentCurrency;
        }
    }
}

```

Лістинг класу “BanksEventArgs”:

```

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

namespace ClassLibraryBank
{
    public class BanksEventArgs
    {
        public string Message { get; }

        public BanksEventArgs(string message)
        {
            Message = message;
        }
    }
}

```

Лістинг Program.cs:

```

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

namespace ConsoleAppBank
{
    class Program
    {
        static void DisplayMessage(AutomatedTellerMachine sender, OperationsBankEventArgs
e)
        {
            Console.WriteLine($"Сумма транзакції: {e.Sum} USD ({e.EquivalentUAN} UAN)");
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine(e.Message);
            Console.ResetColor();
        }
        static string CheckString()
        {
            string input;
            do
            {
                input = Console.ReadLine();
                Console.ForegroundColor = ConsoleColor.Red;
                if (string.IsNullOrEmpty(input))
                {
                    Console.WriteLine(" Помилка введення значення. Будь-ласка, повторіть
введення ще раз!");
                    Console.ResetColor();
                    Console.Write("Введіть значення ще раз: ");
                }
                Console.ResetColor();
            } while (string.IsNullOrEmpty(input));
            return input;
        }
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.Unicode;
            Console.InputEncoding = System.Text.Encoding.Unicode;
            System.Globalization.CultureInfo customCulture =
(System.Globalization.CultureInfo)
            System.Threading.Thread.CurrentThread.CurrentCulture.Clone();
            customCulture.NumberFormat.NumberDecimalSeparator = ".";

```

```

System.Threading.Thread.CurrentThread.CurrentCulture = customCulture;
Console.Title = "Лабораторна робота №1 Божко К. Д , BT-22-2";

float CheckFloat()
{
    bool f;
    float x;
    do
    {
        f = float.TryParse(Console.ReadLine(), out x);
        Console.ForegroundColor = ConsoleColor.Red;
        if (f == false)
        {
            Console.WriteLine(" Помилка введення значення. Будь-ласка, по-
вторіть введення ще раз!");
            Console.ResetColor();
            Console.Write("Введіть значення ще раз: ");
        }
        Console.ResetColor();
    } while (!f);
    return x;
}

int CheckInt()
{
    bool f;
    int x;
    do
    {
        f = int.TryParse(Console.ReadLine(), out x);
        Console.ForegroundColor = ConsoleColor.Red;
        if (f == false)
        {
            Console.WriteLine(" Помилка введення значення. Будь-ласка, по-
вторіть введення ще раз!");
            Console.ResetColor();
            Console.Write("Введіть значення ще раз: ");
        }
        Console.ResetColor();
    } while (!f);
    return x;
}

Account[] account =
{
    new Account ("Kyrylo", "Bozhko ", "3333", 1111, 19000),
    new Account ("Kyrylo", "Bozhko ", "5375 4114 1883 1397", 1111, 16000),
    new Account ("Renat", "Kramarovsky", "5335 4141 2596 1007", 2222,
11200),
    new Account ("Oleksii", "Kolesnyk", "5335 4141 2596 1007", 3333,
12000),
    new Account("Danylo", "Govorun", "5956 4114 2612 9436", 4444, 9000),
    new Account ("Nikita ", "Motytskyi", "5849 4141 2596 1000", 5555,
20000),
};

AutomatedTellerMachine ATM = new AutomatedTellerMachine(100000000000,
"#123456789", "вул. Чуднівська, 103");
Bank bank = new Bank("Mono24", 1);

bank.Notifications += PrintInfo;
autification:
int user;
bool flag = false;
do
{
    Console.WriteLine("Введіть номер карти: "); string card = CheckString();
    Console.WriteLine("Введіть пароль: "); int pin = CheckInt();
    flag = bank.Authentication(card, pin, account, out user);
    Console.ReadKey();
}

```

```

    } while (!flag);
    Console.Clear();

    ATM.Notify += DisplayMessage;
    account[user].RegisterHandler(PrintInfo);

    int num;
    do
    {
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("\n----- Головне меню ----- \n");
        Console.ResetColor();
        Console.WriteLine("1.Вивести дані користувача.\n2.Перевірити ба-
ланс.\n3.Зняти кошти.\n4.Покласти кошти.\n5.Перевести кошти.\n6.Вийти з акаунту.\n0.Вихід
з програми");
        Console.Write("Введіть пункт меню: ");
        num = CheckInt();
        switch (num)
        {
            case 1: Console.Clear(); account[user].PrintInfo(); break;
            case 2: Console.Clear(); account[user].PrintBalance(); break;
            case 3: Console.Clear(); Console.Write("Введіть суму зняття: ");
float take = CheckFloat(); ATM.Take(take, account, user); break;
            case 4:
            {
                Console.Clear();

                // Выбор валюты
                Console.WriteLine("Виберіть валюту:\n1. Гривні (UAH)\n2. Дол-
лари (USD)");

                int currencyChoice = CheckInt();
                if (currencyChoice == 1)
                {
                    ATM.SetCurrency(AutomatedTellerMachine.Currency.UAH);
                }
                else if (currencyChoice == 2)
                {
                    ATM.SetCurrency(AutomatedTellerMachine.Currency.USD);
                }

                Console.WriteLine($"Поточна валюта: {ATM.GetCurrency()}");

                Console.WriteLine("Введіть суму поповнення: ");
                float put = CheckFloat();

                if (ATM.GetCurrency() == AutomatedTellerMachine.Currency.UAH)
                {
                    ATM.Put(put, account, user);
                }
                else if (ATM.GetCurrency() ==
AutomatedTellerMachine.Currency.USD)
                {
                    ATM.PutUsd(put, account, user);
                }
            }
            break;

            case 5:
            {
                Console.Clear();
                Console.Write("Введіть суму поповнення картки: ");
                float sum = CheckFloat();
                Console.Write("Введіть номер картки: ");

```



```

        string cardNumber = CheckString();
        ATM.CardTransfer(sum, cardNumber, account, user);
    }
    break;
case 6: Console.Clear(); goto autification;
case 7:
{
    Console.Clear();
    Console.WriteLine("Виберіть валюту:\n1. Гривні (UAH)\n2. Дол-
лари (USD)");

    int currencyChoice = CheckInt();
    if (currencyChoice == 1)
    {
        ATM.SetCurrency(AutomatedTellerMachine.Currency.UAH);
    }
    else if (currencyChoice == 2)
    {
        ATM.SetCurrency(AutomatedTellerMachine.Currency.USD);
    }
    Console.WriteLine($"Валюта змінена на {ATM.GetCurrency()}");
}
break;
}
} while (num != 0);

void DisplayMessage(AutomatedTellerMachine sender, OperationsBankEventArgs e)
{
    Console.WriteLine($"Сумма транзакції: {e.Sum}UAN");
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine(e.Message);
    Console.ResetColor();
}

void PrintInfo(string message)
{
    Console.ForegroundColor = ConsoleColor.DarkGreen;
    Console.WriteLine(message);
    Console.ResetColor();
    Console.ReadKey();
}
}
}
}
}

```

Результат виконання роботи:

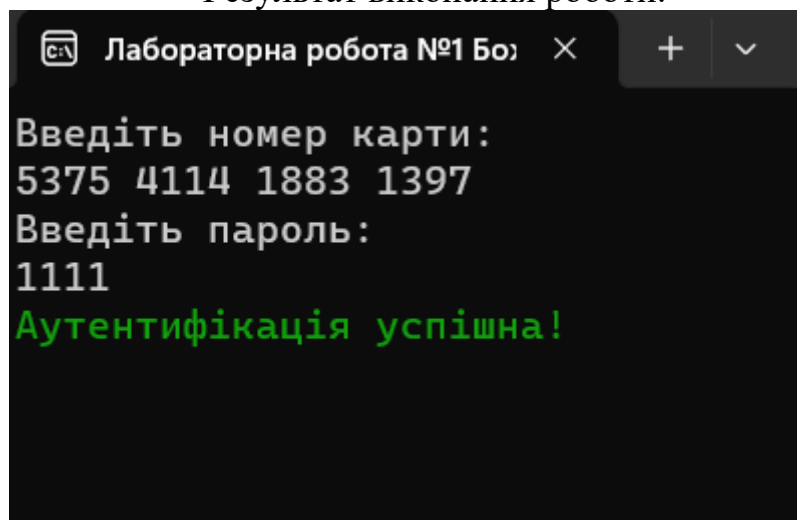
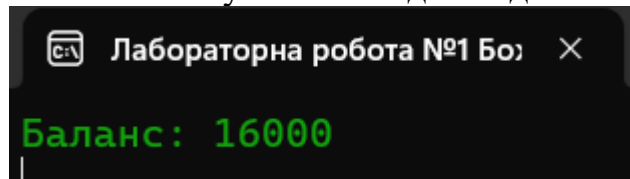


Рис.1.1. Результат аутентифікації

```
Ім'я: Kyrylo  
Прізвище: Bozhko  
Номер картки: 5375 4114 1883 1397  
Баланс: 16000
```

Рис1.2. Результат виведення даних



```
Лабораторна робота №1 Бон X  
Баланс: 16000
```

Рис.1.3. Результат перевірки балансу

```
Введіть суму зняття: 1000  
Сумма транзакції: 1000UAN  
1000 UAN було знято з рахунку  
  
----- Головне меню -----  
  
1.Вивести дані користувача.  
2.Перевірити баланс.  
3.Зняти кошти.  
4.Покласти кошти.  
5.Перевести кошти.  
6.Вийти з акаунту.  
0.Вихід з програми  
Введіть пункт меню: |
```

Рис.1.4. Результат зняття коштів

```

Виберіть валюту:
1. Гривні (UAN)
2. Доллари (USD)
2
Поточна валюта: USD
Введіть суму поповнення:
155
Сумма транзакції: 5890UAN
Поповнено рахунок на 155 USD (5890 UAN) в доларах.

----- Головне меню -----

1.Вивести дані користувача.
2.Перевірити баланс.
3.Зняти кошти.
4.Покласти кошти.
5.Перевести кошти.
6.Вийти з акаунту.
0.Вихід з програми
Введіть пункт меню: |

```

Рис.1.5. Результат роботи зарахування коштів на картку (можливо обрати валюту, якою буде поповнюватися рахунок, якщо поповнювати доларами, тоді сума буде конвертуватися по курсу гривні)

Лістинг Windows Forms:

```

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 ClassLibraryBank;

namespace WindowsFormsBank
{
    public partial class Form1 : Form
    {
        Account[] account =
            {
                new Account ("Kyrylo", "Bozhko ", "5375 4114 1883 1397", 1111, 18200),
                new Account ("Kyrylo", "Bozhko ", "3333", 1111, 18200),
                new Account ("Renat", "Kramarovsky", "5335 4141 2596 1007", 2222,
11200),
                new Account ("Oleksii", "Kolesnyk", "5335 4141 2596 1007", 3333,
12000),
                new Account ("Danylo", "Govorun", "5956 4114 2612 9436", 4444, 9000),
                new Account ("Nikita ", "Motytskyi", "5849 4141 2596 1000", 5555,
20000),
            }
    }
}

```

```

    };
    AutomatedMachine ATM = new AutomatedMachine(10000000000, "#123456789", "вул. Чуд-
нівська, 103");
    Bank bank = new Bank("MONO24", 1);
    int user;
    public Form1()
    {

        InitializeComponent();
        groupBoxOperations.Visible = false;
        label4.Visible = false;
        textBoxCard.Visible = false;
        TransferCard.Visible = false;

    }
    private void Form1_Load(object sender, EventArgs e)
    {
        if (checkBoxNotify.Checked == true) { ATM.Notify -= DisplayMessage; }
        if (checkBoxNotify.Checked == false) { ATM.Notify += DisplayMessage; }
        bank.Notifications += PrintInfo;
        account[user].RegisterHandler(PrintInfo);
        textBoxPin.PasswordChar = '*';
    }
    float CheckFloat(string text)
    {
        bool f;
        float x;
        do
        {
            f = float.TryParse(text, out x);
            if (f == false)
            {
                MessageBox.Show("Помилка введення значення. Будь-ласка, повторіть
введення ще раз!", "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        } while (!f);
        return x;
    }

    int CheckInt(string text)
    {
        bool f;
        int x;
        do
        {
            f = int.TryParse(text, out x);
            if (f == false)
            {
                MessageBox.Show("Помилка введення значення. Будь-ласка, повторіть
введення ще раз!", "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        } while (!f);
        return x;
    }
    private string CheckString(string text)
    {
        string input;
        do
        {
            input = text;
            if (string.IsNullOrEmpty(input))
            {

```

```

        MessageBox.Show("Помилка введення значення. Будь-ласка, повторіть введення ще раз!", "Помилка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        textBoxCardNum.Clear();
    }
} while (string.IsNullOrEmpty(input));

return input;
}
private void buttonAutification_Click(object sender, EventArgs e)
{
    string card = CheckString(textBoxCardNum.Text);
    int pin = CheckInt(textBoxPin.Text);
    bool fl = bank.Authentication(card, pin, account, out user);

    if (fl == true)
    {
        groupBoxAutification.Visible = false;
        groupBoxOperations.Visible = true;
        textBoxCardNum.Text = "";
        textBoxPin.Text = "";
    }
    if (fl == false)
    {
        textBoxCardNum.Text = "";
        textBoxPin.Text = "";
    }
}

void PrintInfo(string message)
{
    MessageBox.Show($"{message}");
}

private void buttonExit_Click(object sender, EventArgs e)
{
    groupBoxOperations.Visible = false;
    groupBoxAutification.Visible = true;
}

private void buttonTake_Click(object sender, EventArgs e)
{
    float take = CheckFloat(textBoxOperations.Text); ATM.Take(take, account,
user);
    groupBoxButtons.Visible = true;
    labelBalance.Text = $"Баланс: {account[user].Balance}";
}
void DisplayMessage(AutomatedMachine sender, OperationsBankEventArgs e)
{
    MessageBox.Show($"Сумма транзакції: {e.Sum}UAN");
    MessageBox.Show(e.Message);
}

private void buttonTran_Click(object sender, EventArgs e)
{
    label4.Visible = true;
    textBoxCard.Visible = true;
    TransferCard.Visible = true;
    groupBoxButtons.Visible = false;
}

private void TransferCard_Click(object sender, EventArgs e)
{
    float sum = CheckFloat(textBoxOperations.Text);
    string card = CheckString(textBoxCard.Text);
    bool flag = ATM.CardTransfer(sum, card, account, user);
}

```

```

        if (flag == true)
        {
            label4.Visible = false;
            textBoxCard.Visible = false;
            TransferCard.Visible = false;
            groupBoxButtons.Visible = true;
            labelBalance.Text = $"Баланс: {account[user].Balance}";
        }

    }

    private void buttonPut_Click(object sender, EventArgs e)
    {
        float take = CheckFloat(textBoxOperations.Text); ATM.Put(take, account,
user);
        groupBoxButtons.Visible = true;
        labelBalance.Text = $"Баланс: {account[user].Balance}";
    }

    private void buttonInfo_Click(object sender, EventArgs e)
    {
        account[user].PrintInfo();
    }

    private void groupBoxOperations_Enter(object sender, EventArgs e)
    {
    }

    private void label1_Click(object sender, EventArgs e)
    {
    }

    }
}

Класс "Bank"
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClassLibraryBank
{
    public delegate void BankHandler(string message);
    public class Bank
    {
        public delegate void BanksHandler(Bank sender, BanksEventArgs e);
        public event BankHandler Notifications;
        public string BankName { get; private set; }
        public int ListOfATMs { get; set; }

        public Bank(string bankName, int listOfATMs)
        {
            BankName = bankName;
            ListOfATMs = listOfATMs;
        }

        public bool Authentication(string cardNumber, int pin, Account[] accounts, out
int user)
        {
            user = -1;
            for (int i = 0; i < Account.amountUsers; i++)
            {

```

```

        if (accounts[i].CardNumber == cardNumber)
        {
            if (accounts[i].Pin == pin)
            {
                user = i;
                Notifications?.Invoke("Аутентифікація успішна!");
                return true;
            }
            else
            {
                Notifications?.Invoke("Помилка введення даних! Повторіть спробу!");
                return false;
            }
        }
    }

    Notifications?.Invoke("Аккаунт не знайдено! Перевірте номер картки.");
    return false;
}
}
}

```

Результат виконання роботи:

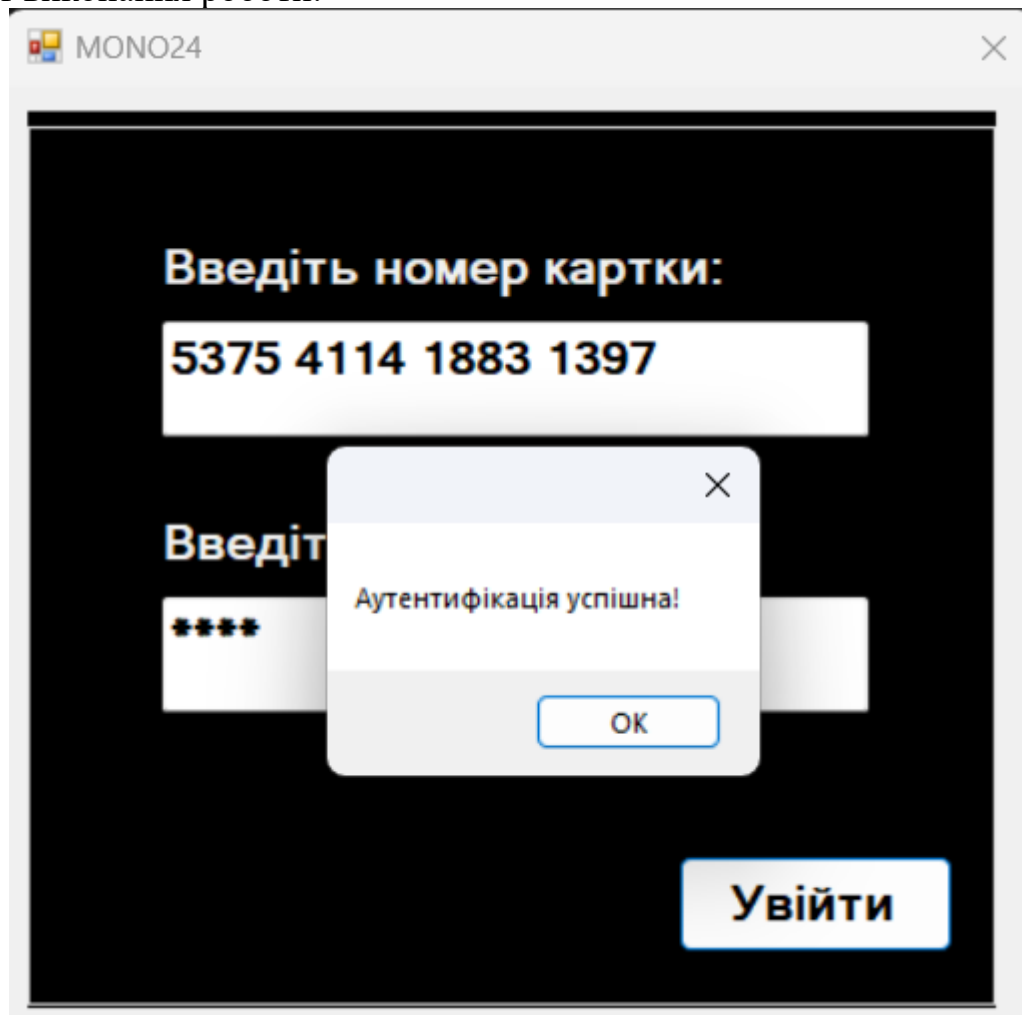


Рис.2.1. Результат аутентифікації

MONO24

×

Введіть суму:

Переглянути  
інформацію

Зняти кошти

Перевести  
кошти

Поповнити  
картку

Вийти

Рис.2.3. Меню



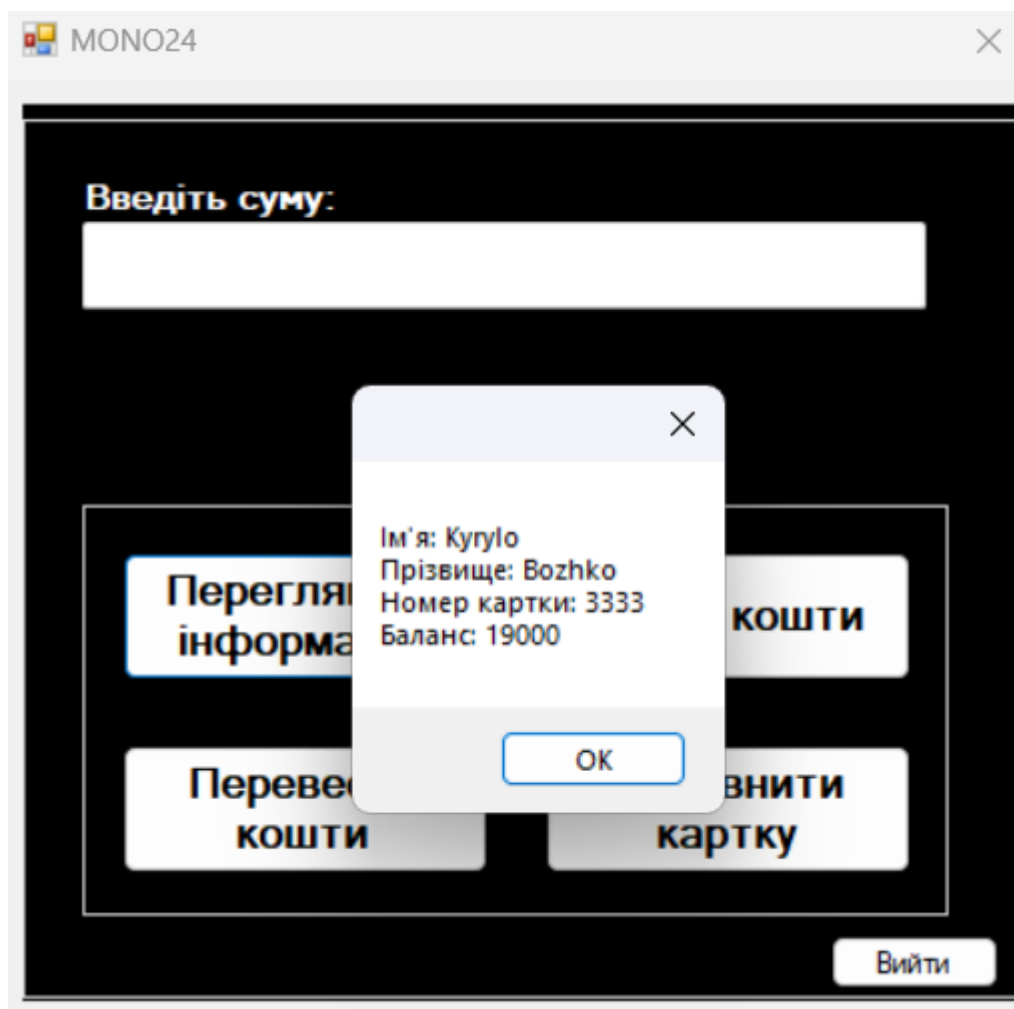


Рис2.4. Перевірка даних користувача

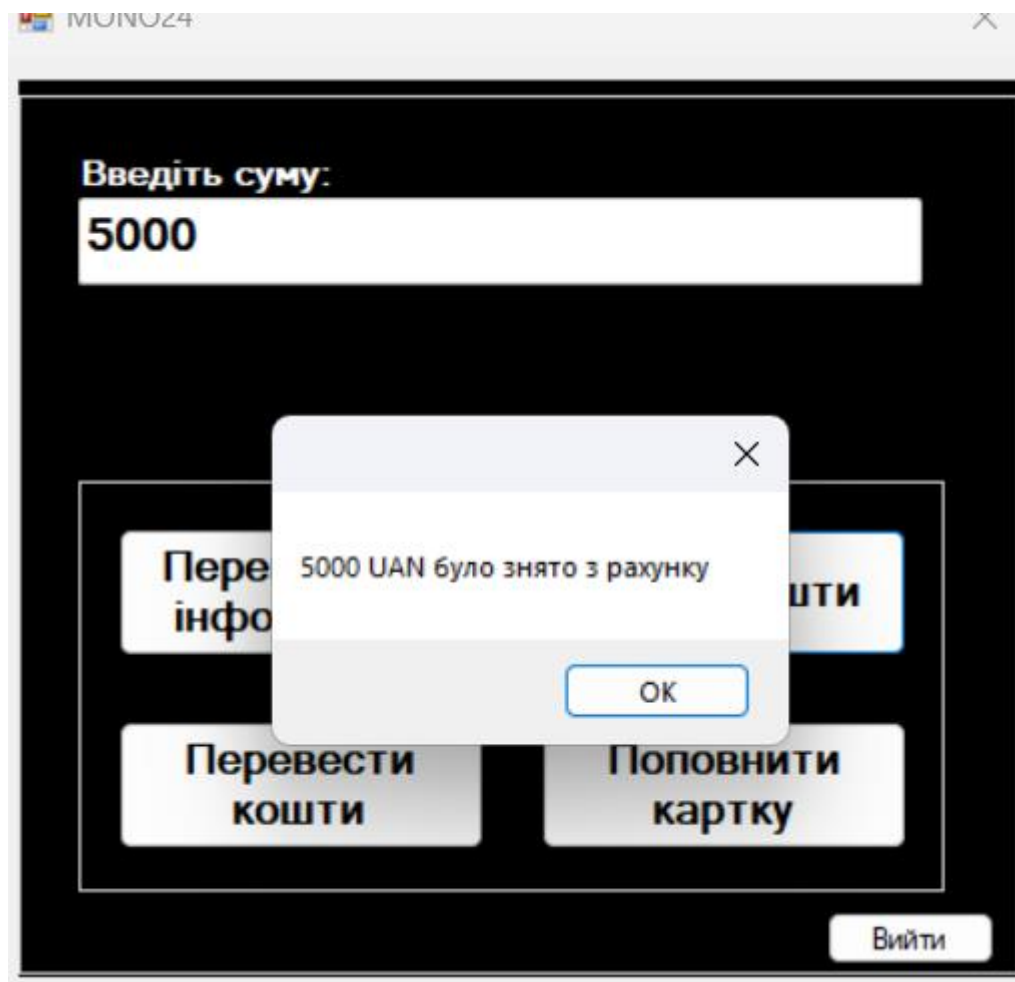


Рис.2.5. Зняття коштів

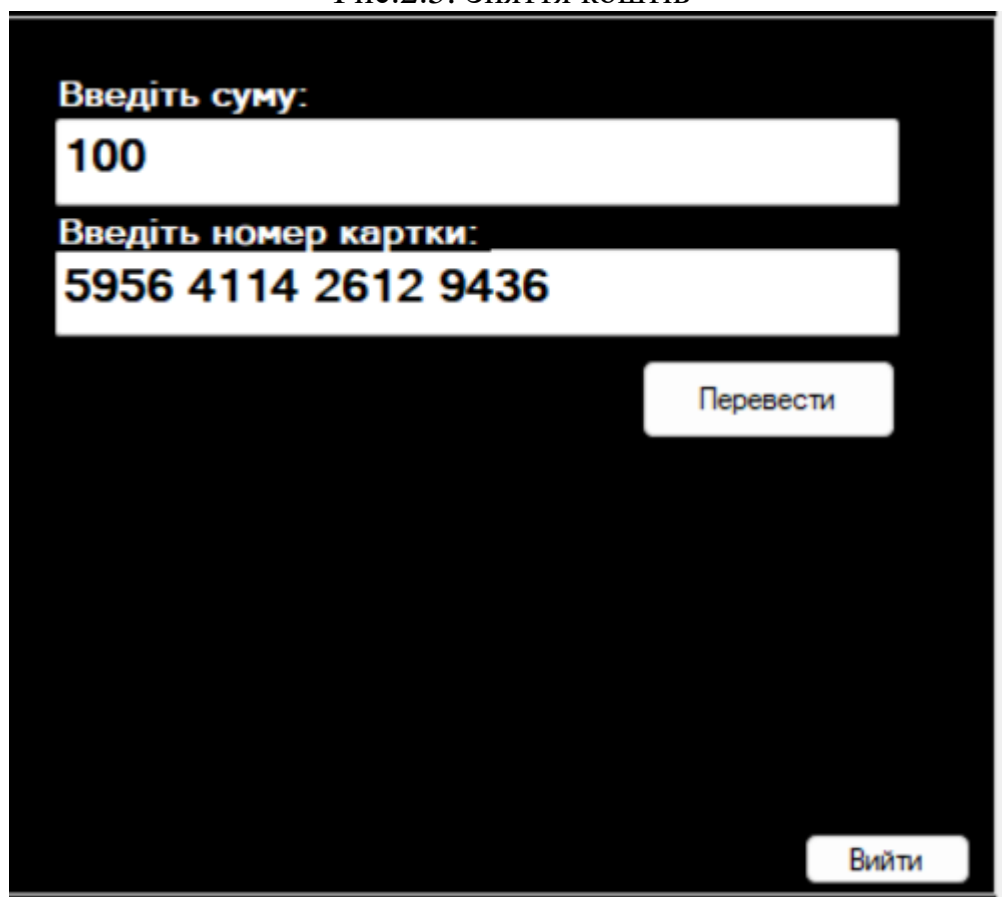


Рис.2.6. Переведення коштів

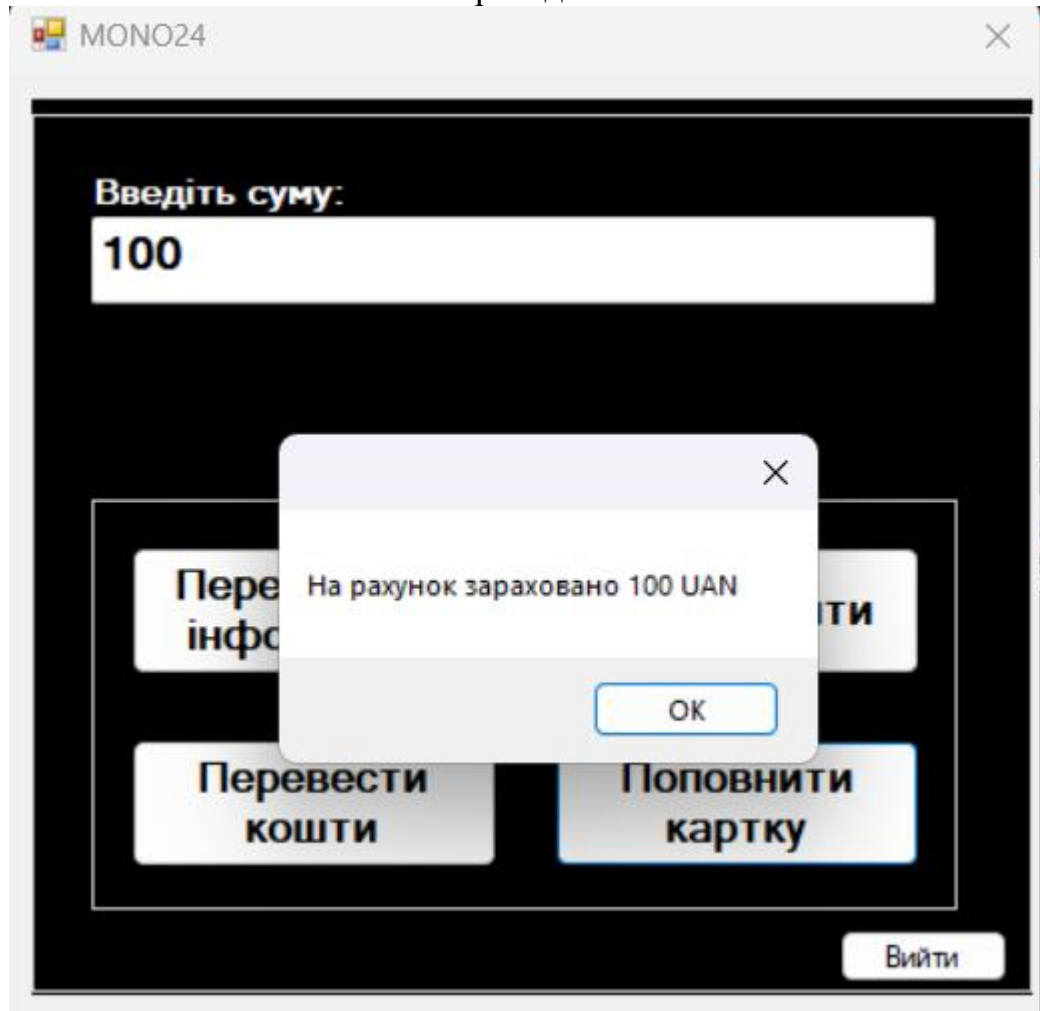


Рис.2.7.Зарахування коштів

**Висновок:** під час виконання лабораторної роботи досліджено процес оголошення та використання делегатів та подій у мові програмування C#.

**Реалізовано програмний код, який доступний за посиланням:**  
<https://gitlab.com/2022-2026/vt-22-2/bozhko/net-core.01-2023-2024.git>