**ЛАБОРАТОРНА РОБОТА № 1**

**Використання делегатів та подій у C#**

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

**Хід роботи:**

**Завдання 2**

**Бібліотека класів:**

Account.cs

namespace ATMClassLibrary

{

public class Account

{

public string cardNumber { get; private set; }

public string firstName { get; private set; }

public string lastName { get; private set; }

public double balance { get; private set; }

public string PIN { get; private set; }

public event EventHandler<AccountAuthenticationEventArgs> AuthenticationEvent;

public event EventHandler<AccountWithdrawalEventArgs> WithdrawalEvent;

public event EventHandler<AccountGetBalanceEventArgs> GetBalanceEvent;

public Account (string cardNumber, string firstName, string lastName, double balance, string PIN)

{

this.cardNumber = cardNumber;

this.firstName = firstName;

this.lastName = lastName;

this.balance = balance;

this.PIN = PIN;

}

public bool authentication (string PIN)

{

string result;

if (this.PIN.Equals(PIN))

result = "Аутентифікація картки успішна! Користувач: " + firstName + " " + lastName + ".";

else

result = "Аутентифікація картки неуспішна! Введено неправильний пін-код";

if (AuthenticationEvent != null)

AuthenticationEvent(this, new AccountAuthenticationEventArgs(result));

return this.PIN.Equals (PIN);

}

public double getBalance ()

{

if (GetBalanceEvent != null)

GetBalanceEvent(this, new AccountGetBalanceEventArgs("Баланс картки: " + this.balance.ToString()));

return this.balance;

}

public bool withdrawal (double balance)

{

if (this.balance < balance)

{

if (WithdrawalEvent != null)

WithdrawalEvent(this, new AccountWithdrawalEventArgs("Недостатньо коштів!"));

return false;

}

this.balance -= balance;

if (WithdrawalEvent != null)

WithdrawalEvent(this, new AccountWithdrawalEventArgs("Операція зняття пройша успішно!"));

return true;

}

public void charging(double balance)

{

this.balance += balance;

return;

}

}

}

AccountAuthenticationEventArgs.cs

namespace ATMClassLibrary

{

public class AccountAuthenticationEventArgs : EventArgs

{

public string result { get; private set; }

public AccountAuthenticationEventArgs (string result)

{

this.result = result;

}

}

}

AccountGetBalanceEventArgs.cs

namespace ATMClassLibrary

{

public class AccountGetBalanceEventArgs : EventArgs

{

public string balance { get; private set; }

public AccountGetBalanceEventArgs(string balance)

{

this.balance = balance;

}

}

}

AccountWithdrawalEventArgs.cs

namespace ATMClassLibrary

{

public class AccountWithdrawalEventArgs : EventArgs

{

public string result { get; private set; }

public AccountWithdrawalEventArgs (string result)

{

this.result = result;

}

}

}

AutomatedTellerMachine.cs

namespace ATMClassLibrary

{

public class AutomatedTellerMachine

{

public double balance { get; private set; }

public string ID { get; private set; }

public string address { get; private set; }

public event EventHandler<AutomatedTellerMachineWithdrawalEventArgs> WithdrawalEvent;

public AutomatedTellerMachine(double balance, string ID, string address)

{

this.balance = balance;

this.ID = ID;

this.address = address;

}

public bool withdrawal(double balance)

{

if (this.balance < balance)

{

if (WithdrawalEvent != null)

WithdrawalEvent(this, new AutomatedTellerMachineWithdrawalEventArgs("Закінчилась готівка! Спробуйте провести операцію в іншому банкоматі!"));

return false;

}

if (WithdrawalEvent != null)

WithdrawalEvent(this, new AutomatedTellerMachineWithdrawalEventArgs("Банкомат успішно видав " + balance.ToString() + "грн!"));

this.balance -= balance;

return true;

}

public void charging(double balance)

{

this.balance += balance;

return;

}

}

}

AutomatedTellerMachineWithdrawalEventArgs.cs

namespace ATMClassLibrary

{

public class AutomatedTellerMachineWithdrawalEventArgs : EventArgs

{

public string result { get; private set; }

public AutomatedTellerMachineWithdrawalEventArgs(string result)

{

this.result = result;

}

}

}

Bank.cs

using System.Net.NetworkInformation;

using System.Security.Principal;

namespace ATMClassLibrary

{

public class Bank

{

public string name { get; private set; }

public List<Account> accounts { get; private set; }

public List<AutomatedTellerMachine> automatedTellerMachines { get; private set; }

public event EventHandler<BankFindATMEventArgs> FindATMEvent;

public event EventHandler<BankFindAccountEventArgs> FindAccountEvent;

public Bank (string name, List<Account> accounts, List<AutomatedTellerMachine> automatedTellerMachines)

{

this.name = name;

this.accounts = accounts;

this.automatedTellerMachines = automatedTellerMachines;

}

public short findATM (string ID)

{

for (short i = 0; i < automatedTellerMachines.Count; i++)

{

if (automatedTellerMachines[i].ID.Equals(ID))

{

if (FindATMEvent != null)

FindATMEvent(this, new BankFindATMEventArgs("Підключення до банкомату з номером " + ID + " успішне!"));

return i;

}

}

if (FindATMEvent != null)

FindATMEvent(this, new BankFindATMEventArgs("Банкомату з номером " + ID + " не знайдено!"));

return -1;

}

public short findAccount (string cardNumber)

{

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

{

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

{

if (FindATMEvent != null)

FindATMEvent(this, new BankFindATMEventArgs("Картка з номером " + cardNumber + " в базі знайдена!"));

return i;

}

}

if (FindATMEvent != null)

FindATMEvent(this, new BankFindATMEventArgs("Картки з номером " + cardNumber + " в базі не знайдено!"));

return -1;

}

public bool withdrawal (string ID, string cardNumber, string PIN, double balance)

{

short iATM = findATM (ID);

if (iATM == -1)

{

return false;

}

short iAccount = findAccount (cardNumber);

if (iAccount == -1)

{

return false;

}

if (!accounts[iAccount].authentication(PIN))

{

return false;

}

if (!accounts[iAccount].withdrawal(balance))

{

return false;

}

if (!automatedTellerMachines[iATM].withdrawal(balance))

{

accounts[iAccount].charging(balance);

return false;

}

return true;

}

public bool charging(string ID, string cardNumber, double balance)

{

short iATM = findATM(ID);

if (iATM == -1)

{

return false;

}

short iAccount = findAccount(cardNumber);

if (iAccount == -1)

{

return false;

}

accounts[iAccount].charging(balance);

automatedTellerMachines[iATM].charging(balance);

return true;

}

public bool transfer(string cardNumber1, string PIN, string cardNumber2, double balance)

{

short iAccount1 = findAccount(cardNumber1);

if (iAccount1 == -1)

{

return false;

}

short iAccount2 = findAccount(cardNumber2);

if (iAccount2 == -1)

{

return false;

}

if (!accounts[iAccount1].authentication(PIN))

{

return false;

}

if (!accounts[iAccount1].withdrawal(balance))

{

return false;

}

accounts[iAccount2].charging(balance);

return true;

}

}

}

BankFindAccountEventArgs.cs

namespace ATMClassLibrary

{

public class BankFindAccountEventArgs : EventArgs

{

public string result { get; private set; }

public BankFindAccountEventArgs(string result)

{

this.result = result;

}

}

}

BankFindATMEventArgs.cs

namespace ATMClassLibrary

{

public class BankFindATMEventArgs : EventArgs

{

public string result { get; private set; }

public BankFindATMEventArgs(string result)

{

this.result = result;

}

}

}

**Консольний проект:**

using ATMClassLibrary;

using System.Text;

Console.OutputEncoding = Encoding.UTF8;

void AccountAuthenticationHandler(object sender, AccountAuthenticationEventArgs e)

{

Console.WriteLine(e.result);

}

void AccountWithdrawalHandler(object sender, AccountWithdrawalEventArgs e)

{

Console.WriteLine(e.result);

}

void AccountGetBalanceHandler(object sender, AccountGetBalanceEventArgs e)

{

Console.WriteLine(e.balance);

}

void ATMWithdrawalHandler(object sender, AutomatedTellerMachineWithdrawalEventArgs e)

{

Console.WriteLine(e.result);

}

void BankFindAccountHandler(object sender, BankFindAccountEventArgs e)

{

Console.WriteLine(e.result);

}

void BankFindATMHandler(object sender, BankFindATMEventArgs e)

{

Console.WriteLine(e.result);

}

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

accounts.Add(new Account("5168123412340000", "Andrew", "Andreev", 3000, "1289"));

accounts.Add(new Account("5168432143219999", "Anna", "Andeeva", 1000, "0038"));

accounts[0].AuthenticationEvent += AccountAuthenticationHandler;

accounts[0].WithdrawalEvent += AccountWithdrawalHandler;

accounts[0].GetBalanceEvent += AccountGetBalanceHandler;

accounts[1].AuthenticationEvent += AccountAuthenticationHandler;

accounts[1].WithdrawalEvent += AccountWithdrawalHandler;

accounts[1].GetBalanceEvent += AccountGetBalanceHandler;

List<AutomatedTellerMachine> automatedTellerMachines = new List<AutomatedTellerMachine>();

automatedTellerMachines.Add(new AutomatedTellerMachine(100000, "01", "Kyivska, 63"));

automatedTellerMachines.Add(new AutomatedTellerMachine(10, "02", "Berdychivska, 18"));

automatedTellerMachines[0].WithdrawalEvent += ATMWithdrawalHandler;

automatedTellerMachines[1].WithdrawalEvent += ATMWithdrawalHandler;

Bank bank = new Bank("ProstoBank", accounts, automatedTellerMachines);

bank.FindAccountEvent += BankFindAccountHandler;

bank.FindATMEvent += BankFindATMHandler;

// -----------------------------------------

while (true)

{

Console.WriteLine("Оберіть банкомат");

for (short i = 0; i < bank.automatedTellerMachines.Count; i++)

Console.WriteLine((i + 1).ToString() + ". " + automatedTellerMachines[i].address + "(" + automatedTellerMachines[i].ID + ").");

string ATM\_ID = bank.automatedTellerMachines[short.Parse(Console.ReadLine()) - 1].ID;

while (true)

{

Console.WriteLine("Оберіть операцію");

Console.WriteLine("1. Перегляд балансу карти.");

Console.WriteLine("2. Зняття коштів.");

Console.WriteLine("3. Зарахування коштів на картку.");

Console.WriteLine("4. Перерахування коштів з картки на картку.");

Console.WriteLine("5. Назад.");

short read = short.Parse(Console.ReadLine());

string cardNumber;

string cardNumber2;

string PIN;

short balance;

bool end = false;

switch (read)

{

case 1:

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

bank.accounts[bank.findAccount(Console.ReadLine())].getBalance();

break;

case 2:

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

cardNumber = Console.ReadLine();

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

PIN = Console.ReadLine();

Console.WriteLine("Введіть суму, яку хочете зняти:");

balance = short.Parse(Console.ReadLine());

bank.withdrawal(ATM\_ID, cardNumber, PIN, balance);

break;

case 3:

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

cardNumber = Console.ReadLine();

Console.WriteLine("Введіть суму, яку хочете зарахувати:");

balance = short.Parse(Console.ReadLine());

bank.charging(ATM\_ID, cardNumber, balance);

break;

case 4:

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

cardNumber = Console.ReadLine();

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

PIN = Console.ReadLine();

Console.WriteLine("Введіть номер картки, на яку хочете переразувати кошти:");

cardNumber2 = Console.ReadLine();

Console.WriteLine("Введіть суму, яку хочете зарахувати:");

balance = short.Parse(Console.ReadLine());

bank.transfer(cardNumber, PIN, cardNumber2, balance);

break;

case 5:

end = true;

break;

}

if (end)

break;

}

}

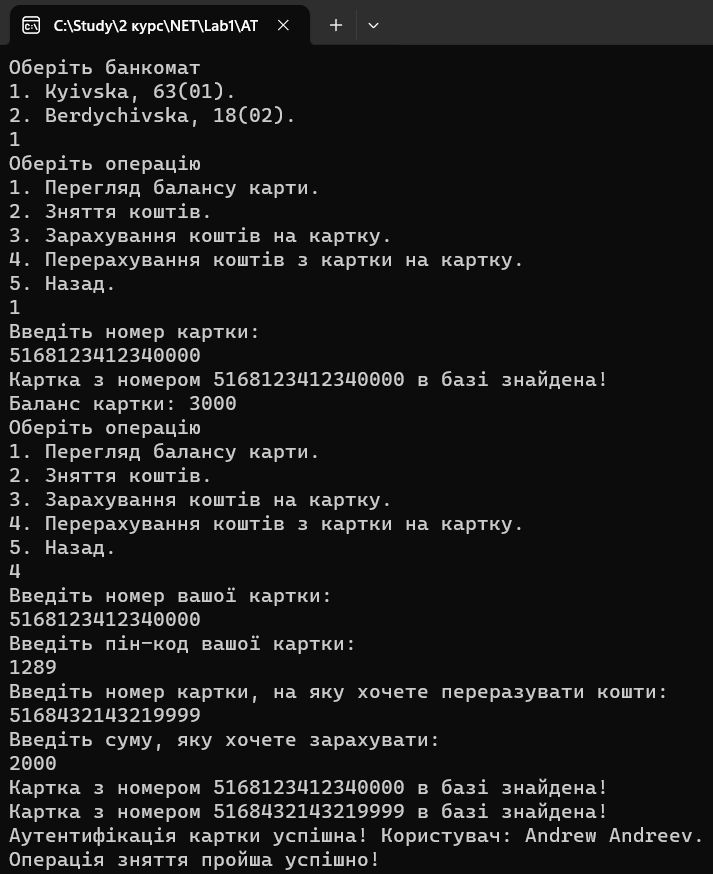


Рис.1. Результат виконання консольного проекту

**Віконний проект:**

using ATMClassLibrary;

namespace WinFormsATM

{

public partial class Form1 : Form

{

public string ATM\_ID;

public string cardNumber;

public string PIN;

public string cardNumber2;

public short balance;

public Bank bank;

public Form1()

{

InitializeComponent();

}

public void load()

{

ATM\_ID = bank.automatedTellerMachines[comboBox1.SelectedIndex].ID;

cardNumber = textBox1.Text;

PIN = textBox2.Text;

cardNumber2 = textBox3.Text;

balance = short.Parse(textBox4.Text);

}

private void Form1\_Load(object sender, EventArgs e)

{

void AccountAuthenticationHandler(object sender, AccountAuthenticationEventArgs e)

{

MessageBox.Show(e.result);

}

void AccountWithdrawalHandler(object sender, AccountWithdrawalEventArgs e)

{

MessageBox.Show(e.result);

}

void AccountGetBalanceHandler(object sender, AccountGetBalanceEventArgs e)

{

MessageBox.Show(e.balance);

}

void ATMWithdrawalHandler(object sender, AutomatedTellerMachineWithdrawalEventArgs e)

{

MessageBox.Show(e.result);

}

void BankFindAccountHandler(object sender, BankFindAccountEventArgs e)

{

MessageBox.Show(e.result);

}

void BankFindATMHandler(object sender, BankFindATMEventArgs e)

{

MessageBox.Show(e.result);

}

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

accounts.Add(new Account("5168123412340000", "Andrew", "Andreev", 3000, "1289"));

accounts.Add(new Account("5168432143219999", "Anna", "Andeeva", 1000, "0038"));

accounts[0].AuthenticationEvent += AccountAuthenticationHandler;

accounts[0].WithdrawalEvent += AccountWithdrawalHandler;

accounts[0].GetBalanceEvent += AccountGetBalanceHandler;

accounts[1].AuthenticationEvent += AccountAuthenticationHandler;

accounts[1].WithdrawalEvent += AccountWithdrawalHandler;

accounts[1].GetBalanceEvent += AccountGetBalanceHandler;

List<AutomatedTellerMachine> automatedTellerMachines = new List<AutomatedTellerMachine>();

automatedTellerMachines.Add(new AutomatedTellerMachine(100000, "01", "Kyivska, 63"));

automatedTellerMachines.Add(new AutomatedTellerMachine(10, "02", "Berdychivska, 18"));

automatedTellerMachines[0].WithdrawalEvent += ATMWithdrawalHandler;

automatedTellerMachines[1].WithdrawalEvent += ATMWithdrawalHandler;

bank = new Bank("ProstoBank", accounts, automatedTellerMachines);

bank.FindAccountEvent += BankFindAccountHandler;

bank.FindATMEvent += BankFindATMHandler;

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

{

comboBox1.Items.Add(automatedTellerMachines[i].address + "(" + automatedTellerMachines[i].ID + ")");

}

comboBox1.SelectedIndex = 0;

}

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

{

load();

bank.accounts[bank.findAccount(cardNumber)].getBalance();

}

private void button2\_Click(object sender, EventArgs e)

{

load();

bank.withdrawal(ATM\_ID, cardNumber, PIN, balance);

}

private void button3\_Click(object sender, EventArgs e)

{

load();

bank.charging(ATM\_ID, cardNumber, balance);

}

private void button4\_Click(object sender, EventArgs e)

{

load();

bank.transfer(cardNumber, PIN, cardNumber2, balance);

}

}

}

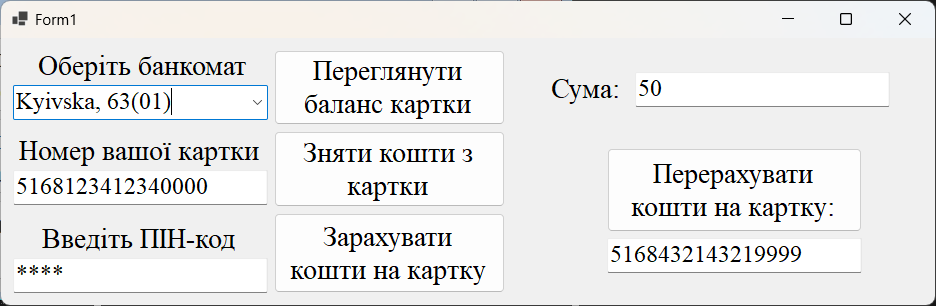


Рис.2. Головне вікно

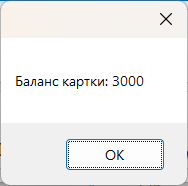
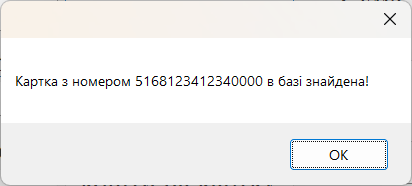


Рис.3. Результат перегляду балансу картки

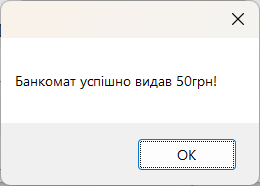
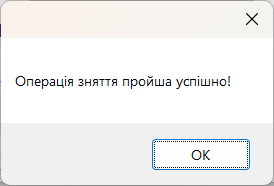
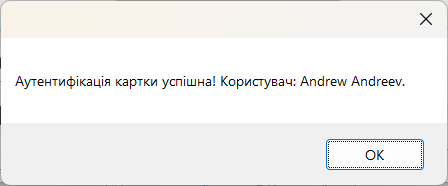
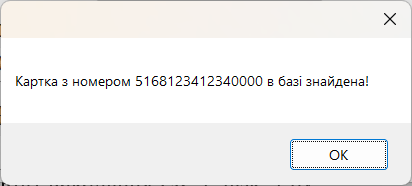
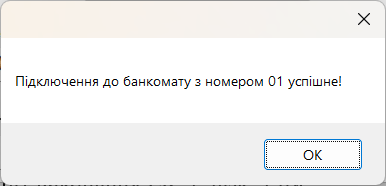


Рис.4. Результат зняття коштів з картки

**Завдання 3**

Репозиторій: <https://github.com/Kochubei-Kostiantyn/DotNetLab1>

***Висновки:*** в ході виконання лабораторної роботи ми навчилися використовувати оголошувати та використовувати делегати та події у мові програмування C#.