Лабораторна робота №1

Тема: Використання делегатів та подій у C#.

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

Виділений час: 12 годин (4 години лабораторних робіт та 8 годин самостійної роботи).

Завдання:

1. Опрацювати теорію:
відеозаписи лекцій №01-02;

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 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");

Також можна відправляти повідомлення на електронну пошту.

Для відправки листів на електронну пошту можна використовувати код:

```
var smtpClient = new SmtpClient("smtp.gmail.com")
{
  Port = 587,
  UseDefaultCredentials = false,
  Credentials = new NetworkCredential("Електронна пошта",
"Пароль"),
  EnableSsl = true,
  DeliveryMethod = SmtpDeliveryMethod.Network
};
smtpClient.Send("Пошта відправника", "Пошта отримувача", "Тема
листа", "Текст листа");
```

Для того, щоб даний код працював потрібно в гугл-акаунті включити доступ
сторонніх додатків (ця операція здійснюється за адресою: https://myaccount.google.com/u/1/lesssecureapps).

Якщо Ви використовуєте даний код, то при виконанні коміту видаліть власний
пароль, який прописується в коді.

При виконанні завдання створіть класи, які забезпечать роботу :
● Account - клас, що представляє картковий рахунок клієнта (властивості: номер
картки, прізвище та ім’я власника, баланс на рахунку і т.д.);
● AutomatedTellerMachine - клас, що моделює роботу банкомата (містить кількість
грошей, які містяться у банкоматі, ідентифікатор банкомата, адресу і т.п.);
● Bank - клас, що представляє банк (назва банку, список банкоматів і т.п.).

Це не повний список класів, за необхідності створюйте інші класи, які Вам будуть
необхідні при реалізації завдання.

3. Запушити виконану роботу у репозиторій на GitHub за назвою DotNetLab1 і
відкрити доступ для викладачів:
● morozov@ztu.edu.ua
● 4ov.ztu@gmail.com

Класи:

Bank:

```
namespace ATMClassLibrary.Entities
{
```

```
    public class Bank
    {
        public string Name { get; private set; }
        private List<AutomatedTellerMachine> atms;
        private List<Account> accounts;

        public Bank(string name)
        {
            Name = name;
            atms = new List<AutomatedTellerMachine>();
            accounts = new List<Account>();
        }

        public void AddATM(AutomatedTellerMachine atm) => atms.Add(atm);
        public void AddAccount(Account account) => accounts.Add(account);

        public bool Authenticate(string cardNumber, int pinCode)
        {
            var account = GetAccount(cardNumber);
            return account != null && account.PIN == pinCode;
        }

        public Account GetAccount(string cardNumber) => accounts.Find(a => a.CardNumber == cardNumber);

        public bool Withdraw(string cardNumber, decimal amount)
        {
            var account = GetAccount(cardNumber);
            if (account != null && account.Balance >= amount)
            {
                account.Balance -= amount;
                return true;
            }
            return false;
        }

        public bool Deposit(string cardNumber, decimal amount)
        {
            var account = GetAccount(cardNumber);
            if (account != null)
            {
                account.Balance += amount;
                return true;
            }
            return false;
        }
    }
}
```

AutomatedTellerMachine:

```
namespace ATMClassLibrary.Entities
{
    public class AutomatedTellerMachine
    {
```

```
        public string Id { get; private set; }
        public string Address { get; private set; }
        public decimal CashAvailable { get; set; }

        public AutomatedTellerMachine(string id, string address, decimal cashAvailable)
        {
            Id = id;
            Address = address;
            CashAvailable = cashAvailable;
        }
    }


}
```

Account:

```
namespace ATMClassLibrary.Entities
{
    public class Account
    {
        public string CardNumber { get; private set; }
        public string Owner { get; private set; }
        public int PIN { get; private set; }
        public decimal Balance { get; set; }

        public Account(string cardNumber, string owner, int pin, decimal balance)
        {
            CardNumber = cardNumber;
            Owner = owner;
            PIN = pin;
            Balance = balance;
        }
    }
}
```