МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Практикум №4

з курсу «Основи розробки програмного забезпечення на платформі Microsoft.NET»

на тему: «Шаблони проектування.
Струткturні шаблони»

Викладач:
Крамар Ю.М.

Виконав:
студент 2 курсу
групи ІП-15 ФІОТ
Костін В.А.

Київ-2023

Комп‘ютерний практикум № 4.

Тема: шаблони проектування, струтктурні шаблони.

Мета: ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

Постановка задачі комп‘ютерного практикуму № 4

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Вивчити породжуючі патерни. Знати загальну характеристику та призначення кожного з них, особливості реалізації кожного з породжуючих патернів та випадки їх застосування.

2) Реалізувати задачу згідно варіанту, запропонованого нижче у вигляді консольного застосування на мові C#. Розробити інтерфейси та класи з застосування одного або декількох патернів. Повністю реалізувати методи, пов‘язані з реалізацією обраного патерну.

3) Повністю описати архітектуру проекту (призначення методів та класів), особливості реалізації обраного патерну. Для кожного патерну необхідно вказати основні класи та їх призначення,

4) Навести UML-діаграму класів

5) Звіт повинен містити:

a. обґрунтування обраного патерну (чому саме він);

b. опис архітектури проекту (призначення методів та класів);

c. UML-діаграму класів

d. особливості реалізації обраного патерну

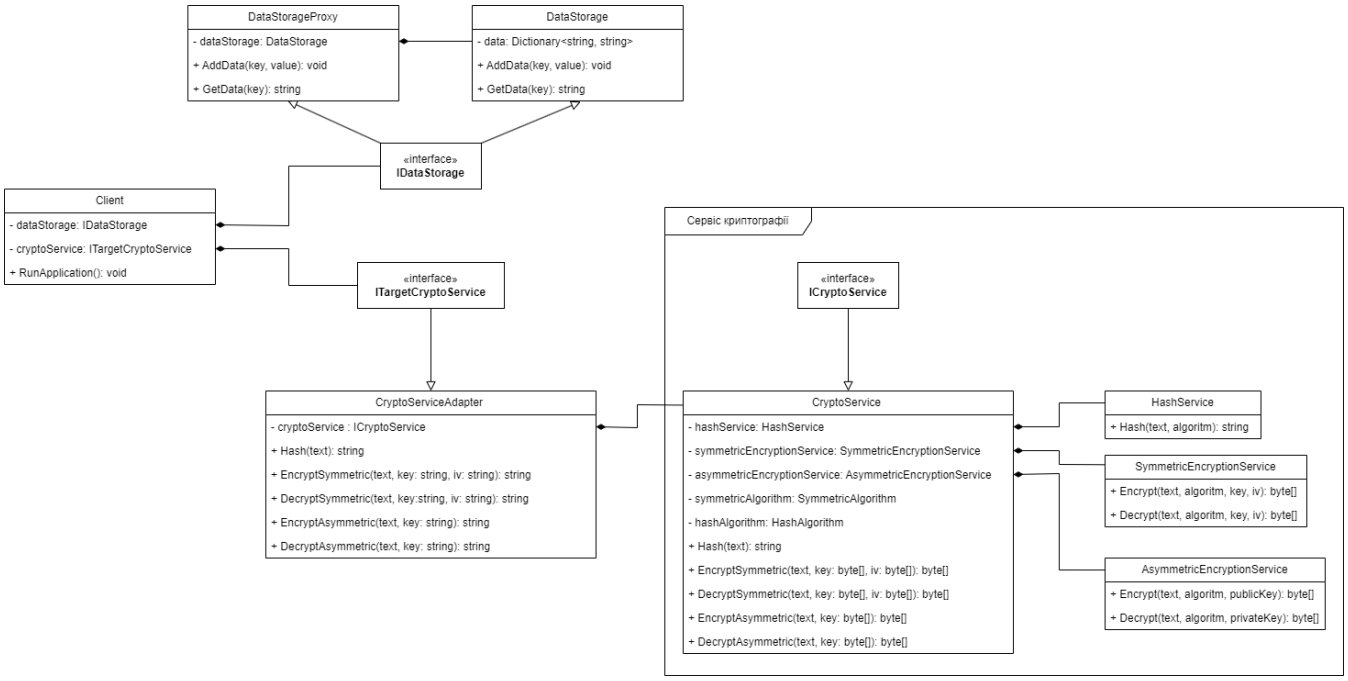
e. текст програмного коду

f. скріншоти результатів виконання.

2) Реалізувати задачу криптографічного захисту даних з використанням бібліотеки System.Security.Cryptography. Система криптографічного захисту повинна реалізовувати шифрування даних с використанням обраної комбінації алгоритмів хешування, симетричного та асиметричного

шифрування. Необхідно реалізувати й дешифрування (ключі необхідно запитувати у користувача).

UML-діаграма класів:



Архітектура програми

Назва класу: HashService

Призначення: сервіс, що надає засоби хешування данних.

Опис методів:

string Hash(string text, HeshAlgoritm algoritm) – метод, що обчислює хеш-ключ для введеного рядку за допомогою введеного алгоритму.

Назва класу: SymmetricEncryptionService

Призначення: сервіс, що надає засоби симетричного шифрування даних.

Опис методів:

byte[] Encrypt(string text, SymmetricAlgorithm algorithm, byte[] key, byte[] iv) – метод, що шифрує введені дані за ключем та вектором ініціалізації.

byte[] Decrypt(string encryptedText, SymmetricAlgorithm algorithm, byte[] key, byte[] iv) – метод, що дешифрує введені дані за ключем та вектором ініціалізації.

Назва класу: AsymmetricEncryptionService

Призначення: сервіс, що надає засоби асиметричного шифрування даних.

Опис методів:

byte[] Encrypt(string text, AsymmetricAlgorithm algorithm, byte[] publicKey) – метод, що шифрує введені дані за публічним ключем.

byte[] Decrypt(string encryptedText, AsymmetricAlgorithm algorithm, byte[] privateKey) – метод, що дешифрує введені дані за приватним ключем та вектором ініціалізації.

Назва класу: CryptoService

Призначення: сервіс, що надає засоби хешування, синхронного та асинхронного шифрування. Клас є фасадом, так як він групує сервіси в один інтерфейс та має вже встановлені алгоритми хешування та шифрування, щоб не делегувати вибір алгоритмів користувачу.

Опис властивостей:

HashService _hashService – сервіс хешування даних.

SymmetricEncryptionService _symmetricEncryptionService – сервіс симетричного шифрування даних.

AsymmetricEncryptionService _asymmetricEncryptionService – сервіс асиметричного шифрування даних.

SymmetricAlgorithm _symmetricAlgorithm – алгоритм симетричного шифрування даних.

HashAlgorithm _hashAlgorithm – алгоритм хешування даних.

Опис методів:

string Hash(string text) – метод, що робить хешування введених даних.

byte[] EncryptSymmetric(string text, byte[] key, byte[] iv) – метод, що шифрує введені дані за ключем та вектором ініціалізації.

byte[] DecryptSymmetric(string encryptedText, byte[] key, byte[] iv) – метод, що дешифрує введені дані за ключем та вектором ініціалізації.

byte[] EncryptAsymmetric(string text, byte[] publicKey) – метод, що шифрує введені дані за публічним ключем.

byte[] DecryptAsymmetric(string encryptedText, byte[] privateKey) – метод, що дешифрує введені дані за приватним ключем.

Назва класу: CryptoServiceAdapter

Призначення: адаптер сервісу криптографії, що спрощує інтерфейс основного сервісу та приводить цього до цільового інтерфейсу.

Опис властивостей:

CryptoService _cryptoService – сервіс криптографії, інтерфейс якого адаптується.

Опис методів:

string Hash(string text) – метод, що робить хешування введених даних.

string EncryptSymmetric(string text, string keyStr, string ivStr)– метод, що шифрує введені дані за ключем та вектором ініціалізації.

string DecryptSymmetric(string encryptedText, string keyStr, string ivStr) – метод, що дешифрує введені дані за ключем та вектором ініціалізації.

string EncryptAsymmetric(string text, string publicKeyStr) – метод, що шифрує введені дані за публічним ключем.

string DecryptAsymmetric(string encryptedText, string privateKeyStr) – метод, що дешифрує введені дані за приватним ключем.

Назва класу: DataStorage

Призначення: сховище даних, яке зберігає зашифровані дані за хеш ключем.

Опис властивостей:

Dictionary<string, string> _data – словник, в якому зберігаються данні.

Опис методів:

void AddData(string key, string value) – метод для введення даних в сховище даних за ключем і значенням.

string? GetData(string key) – метод, для пошуку значення в сховищі даних за ключем.

Назва класу: DataStorageProxy

Призначення: замісник основного сховища даних; посередник між сховищем даних та внутрішньою системою.

Опис властивостей:

DataStorage _dataStorage – основне сховище даних.

Опис методів:

void AddData(string key, string value) – метод, що робить хешування введених даних.

string? GetData(string key) – метод, що шифрує введені дані за ключем та вектором ініціалізації.

Назва класу: Client

Призначення: клієнт, який користується сервісом криптографії та зберіганням даних в сховищі даних.

Опис властивостей:

DataStorageProxy _dataStorage – сховище даних.

CryptoServiceAdapter _cryptoService – сервіс криптографії.

Опис методів:

void RunApplication() – метод, що розпочинає роботу системи.

Шаблони

В данній лабораторній роботі було використано три структурні паттерни: фасад, адаптер та посередник(Proxy).

Використання фасаду обумовлено необхідністю згрупувати усі сервіси, що стосуються криптографії, хешування та шифрування даних, в один інтерфейс та встановлення алгоритмів хешування та шифрування.

Так як за нашим сценарієм сервіси криптографії є стороннім сервісом, в нашій внутрішній системі використання цього інтерфейсу може бути складним. Тому виникає потреба в створенні адаптеру – засобу, що полегшує інтерфейс стороннього сервісу до більш прийнятного вигляду для внутрішньої системи.

Для валідації введених даних у сховище даних доцільно створити замісник основного сховища даних. Основною задачею цього замісника є перевірка вхідних параметрів на коректність(перевірка на дублікат та правильний формат ключів) перед безпосереднім введенням даних в основне сховище даних.

Реалізація шаблонів

Паттерн фасад був реалізований за допомогою інтерфейсу ICryptoService та класу CryptoService, який його імплементує. Клас CryptoService поєднує функціональність HashService, SymmetricEncryptionService та AymmetricEncryptionService та делегує на себе вибір алгоритмів для цих сервісів, щоб користувачу надати інтерфейс тільки введення даних та ключів для шифрування. Як я вже зазначив, за сценарієм сервіс криптографії є стороннім сервісом. Тому я виокремив всі сервіси, що стосують криптографії в окрему бібліотеку класів.

Клас адаптер реалізовано за допомогою інтерфейсу сторонього сервісу криптографії IcryptoService, який виконує роль Adaptee, цільового

інтерфейсу для внутрішьої системи ITargetCryptoService який виконує роль Target, та роль адаптеру виконує роль клас CryptoServiceAdapter, який має зв’язок-композицію із класом CryptoService, який адаптується під внутрішнью систему. Основною мотивацією для адаптації є полегшення формату вхідних та вихідних параметрів: CryptoService на вхід приймає ключи та повертає шифр у форматі byte[], а CryptoServiceAdapter – у форматі string і делегує на себе конвертацію рядка в байти та навпаки. Після конвертації рядка в байти адаптер виклає методи CryptoService та результат конвертує з байтів в рядок.

Паттерн замісник був реалізований за допомогою надання сховищу даних та посереднику спільного інтерфейсу та побудування між ними зв’язку композиції. Якщо основне сховище даних має словник у властивостях, то посередник має основне сховище даних у властивостях. В своїх методах посередник робить валідацію вхідних даних та вносить данні в основне сховище даних.

Код програми

Код програми можна подивитись на репозиторії за посиланням:

https://github.com/VadimkaKostin/.Net

HashService.cs

```
public class HashService
    {
        public string Hash(string text, HashAlgorithm algorithm)
        {
            byte[] hashBytes =
algorithm.ComputeHash(Encoding.UTF8.GetBytes(text));
            return BitConverter.ToString(hashBytes);
        }
    }
```

SymmetricEncryptionService.cs

```
public class SymmetricEncryptionService
    {
        public byte[] Encrypt(string text, SymmetricAlgorithm algorithm, byte[]
key, byte[] iv)
        {
            byte[] encryptedBytes;

            using (ICryptoTransform encryptor = algorithm.CreateEncryptor(key,
rgbIV: iv))
            {
```

```
                byte[] plainBytes = Encoding.UTF8.GetBytes(text);
                encryptedBytes = encryptor.TransformFinalBlock(plainBytes, 0,
plainBytes.Length);
            }

            return encryptedBytes;
        }

        public byte[] Decrypt(string encryptedText, SymmetricAlgorithm algorithm,
byte[] key, byte[] iv)
        {
            byte[] decryptedBytes;

            using (ICryptoTransform decryptor = algorithm.CreateDecryptor(key,
rgbIV: iv))
            {
                byte[] encryptedBytes = Convert.FromBase64String(encryptedText);
                decryptedBytes = decryptor.TransformFinalBlock(encryptedBytes, 0,
encryptedBytes.Length);
            }

            return decryptedBytes;
        }
    }
```

AsymmetricEncryptionService.cs

```
public class AsymmetricEncryptionService
    {
        public byte[] Encrypt(string text, byte[] publicKey)
        {
            byte[] encryptedBytes;

            using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
            {
                rsa.ImportRSAPublicKey(publicKey, out _);
                encryptedBytes = rsa.Encrypt(Encoding.UTF8.GetBytes(text), true);
            }

            return encryptedBytes;
        }

        public byte[] Decrypt(string encryptedText, byte[] privateKey)
        {
            byte[] decryptedBytes;

            using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
            {
                rsa.ImportRSAPrivateKey(privateKey, out _);
                decryptedBytes =
rsa.Decrypt(Convert.FromBase64String(encryptedText), true);
            }

            return decryptedBytes;
        }
    }
```

CryptoService.cs

```
public class CryptoService : ICryptoService
    {
        private readonly HashService _hashService;
        private readonly SymmetricEncryptionService _symmetricEncryptionService;
```

```
        private readonly AsymmetricEncryptionService
_asymmetricEncryptionService;
        private readonly SymmetricAlgorithm _symmetricAlgorithm;
        private readonly HashAlgorithm _hashAlgorithm;
        public CryptoService()
        {
            _hashService = new HashService();
            _symmetricEncryptionService = new SymmetricEncryptionService();
            _asymmetricEncryptionService = new AsymmetricEncryptionService();
            _symmetricAlgorithm = new AesCryptoServiceProvider();
            _hashAlgorithm = new HMACMD5();
        }
        public string Hash(string text)
        {
            return _hashService.Hash(text, _hashAlgorithm);
        }
        public byte[] EncryptSymmetric(string text, byte[] key, byte[] iv)
        {
            return _symmetricEncryptionService.Encrypt(text, _symmetricAlgorithm,
key, iv);
        }
        public byte[] DecryptSymmetric(string encryptedText, byte[] key, byte[]
iv)
        {
            return _symmetricEncryptionService.Decrypt(encryptedText,
_symmetricAlgorithm, key, iv);
        }
        public byte[] EncryptAsymmetric(string text, byte[] publicKey)
        {
            return _asymmetricEncryptionService.Encrypt(text, publicKey);
        }
        public byte[] DecryptAsymmetric(string encryptedText, byte[] privateKey)
        {
            return _asymmetricEncryptionService.Decrypt(encryptedText,
privateKey);
        }
    }
}
```

DataStorage.cs

```
public class DataStorage : IDataStorage
    {
        private readonly Dictionary<string, string> _data;

        public DataStorage()
        {
            _data = new Dictionary<string, string>();
        }

        public void AddData(string key, string value)
        {
            _data[key] = value;
        }

        public string? GetData(string key)
        {
            return _data.ContainsKey(key) ? _data[key] : null;
        }
    }
```

DataStorageProxy.cs

```
public class DataStorageProxy : IDataStorage
    {
        private readonly IDataStorage _dataStorage;

        public DataStorageProxy()
        {
            _dataStorage = new DataStorage();
        }

        public void AddData(string key, string value)
        {
            if(_dataStorage.GetData(key) != null)
            {
                throw new ArgumentException("The record with this key is already
present in the data store.");
            }

            _dataStorage.AddData(key, value);
        }

        public string? GetData(string key)
        {
            if(string.IsNullOrEmpty(key))
            {
                throw new ArgumentException("Key is required");
            }

            return _dataStorage.GetData(key);
        }
    }
```

CryptoServiceAdapter.cs

```
public class CryptoServiceAdapter : ITargetCryptoService
    {
        private readonly ICryptoService _cryptoService;

        public CryptoServiceAdapter()
        {
            _cryptoService = new CryptoService();
        }

        public string Hash(string text)
        {
            return _cryptoService.Hash(text);
        }

        public string EncryptSymmetric(string text, string keyStr, string ivStr)
        {
            byte[] key = Encoding.UTF8.GetBytes(keyStr);
            byte[] iv = Encoding.UTF8.GetBytes(ivStr);

            byte[] encriptedData = _cryptoService.EncryptSymmetric(text, key,
iv);

            return Convert.ToBase64String(encriptedData);
        }

        public string DecryptSymmetric(string encryptedText, string keyStr,
string ivStr)
        {
            byte[] key = Encoding.UTF8.GetBytes(keyStr);
            byte[] iv = Encoding.UTF8.GetBytes(ivStr);
```

```
            byte[] decriptedData = _cryptoService.DecryptSymmetric(encryptedText,
key, iv);

            return Encoding.UTF8.GetString(decriptedData);
        }

        public string EncryptAsymmetric(string text, string publicKeyStr)
        {
            byte[] publicKey = Encoding.UTF8.GetBytes(publicKeyStr);
            byte[] encriptedData = _cryptoService.EncryptAsymmetric(text,
publicKey);
            return Convert.ToBase64String(encriptedData);
        }

        public string DecryptAsymmetric(string encryptedText, string
privateKeyStr)
        {
            byte[] privateKey = Encoding.UTF8.GetBytes(privateKeyStr);
            byte[] decriptedData =
_cryptoService.DecryptAsymmetric(encryptedText, privateKey);
            return Encoding.UTF8.GetString(decriptedData);
        }
    }
```

Client.cs

```
public class Client
    {
        private readonly IDataStorage _dataStorage;
        private readonly ITargetCryptoService _cryptoService;

        public Client(IDataStorage dataStorage, ITargetCryptoService
cryptoService)
        {
            this._dataStorage = dataStorage;
            this._cryptoService = cryptoService;
        }

        public void RunApplication()
        {
            Console.Write("Enter key: ");
            string? encriptionKey = Console.ReadLine();

            Console.Write("Enter IV: ");
            string? encriptionIV = Console.ReadLine();

            while (true)
            {
                Console.WriteLine("\nMenu");
                Console.WriteLine("1) Encrypt data");
                Console.WriteLine("2) Decrypt data");
                Console.WriteLine("3) Exit");
                Console.Write("Select an option: ");

                string? option = Console.ReadLine();
                switch (option)
                {
                    case "1":
                        Console.Write("Enter data to encrypt: ");
                        string dataToEncrypt = Console.ReadLine();
                        string encryptedText =
_cryptoService.EncryptSymmetric(dataToEncrypt, encriptionKey, encriptionIV);
```

```
                        string encryptedKey = _cryptoService.Hash(dataToEncrypt);
                        _dataStorage.AddData(encryptedKey, encryptedText);
                        Console.WriteLine("\nData encrypted and stored
successfully!");
                        Console.WriteLine($"Key of data in data store:
{encryptedKey}");
                        break;

                    case "2":
                        Console.Write("Enter key for decryption: ");
                        string? key = Console.ReadLine();
                        string? encryptedData = _dataStorage.GetData(key);
                        string decryptedText =
_cryptoService.DecryptSymmetric(encryptedData, encriptionKey, encriptionIV);
                        Console.WriteLine($"\nDecrypted text: {decryptedText}");
                        break;

                    case "3":
                        return;

                    default:
                        Console.WriteLine("Invalid option selected. Please try
again.");
                        break;
                }
            }
        }
```

Program.cs

```
public static class Program
    {
        public static void Main(string[] args)
        {
            IDataStorage dataStorage = new DataStorageProxy();
            ITargetCryptoService targetCryptoService = new
CryptoServiceAdapter();

            Client client = new Client(dataStorage, targetCryptoService);

            client.RunApplication();
        }
    }
```

Результат роботи програми

Під час роботи програми клієнт входить у систему та вводить ключ та вектор ініціаліхації для шифрування. Далі перед ним з’являється меню, де він обирає опції:

1) Зашифрувати данні
2) Дешифрувати данні
3) Вийти

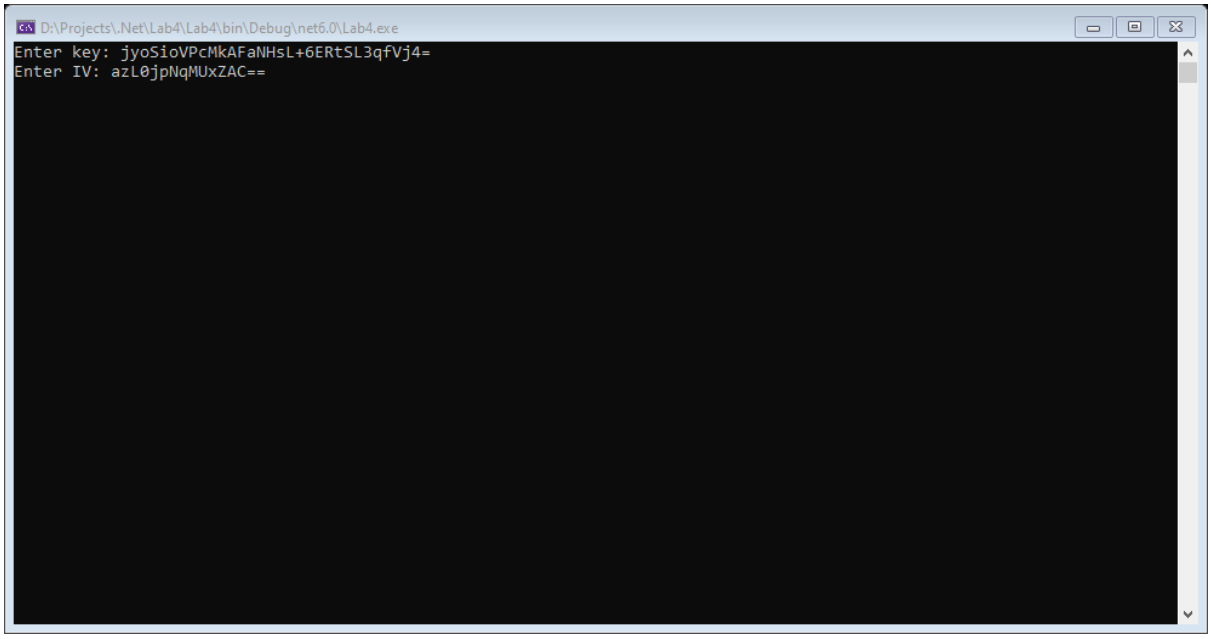


Рисунок 1.1 – введення користувачем ключа та вектору ініціалізації.

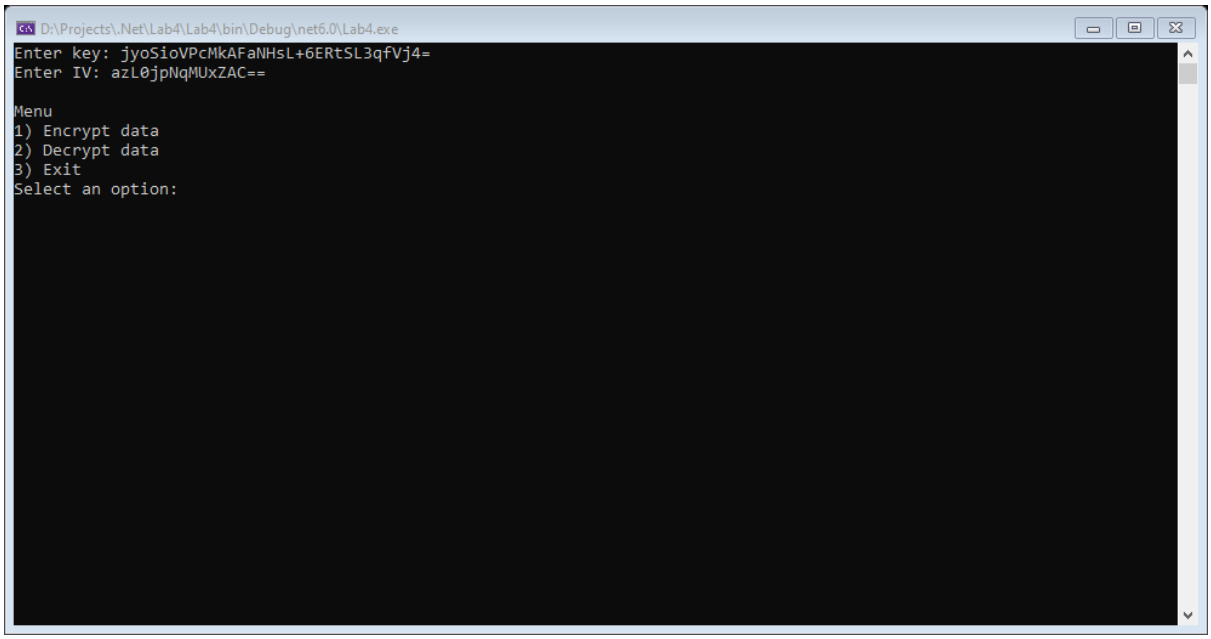


Рисунок 1.2 – запуск меню.

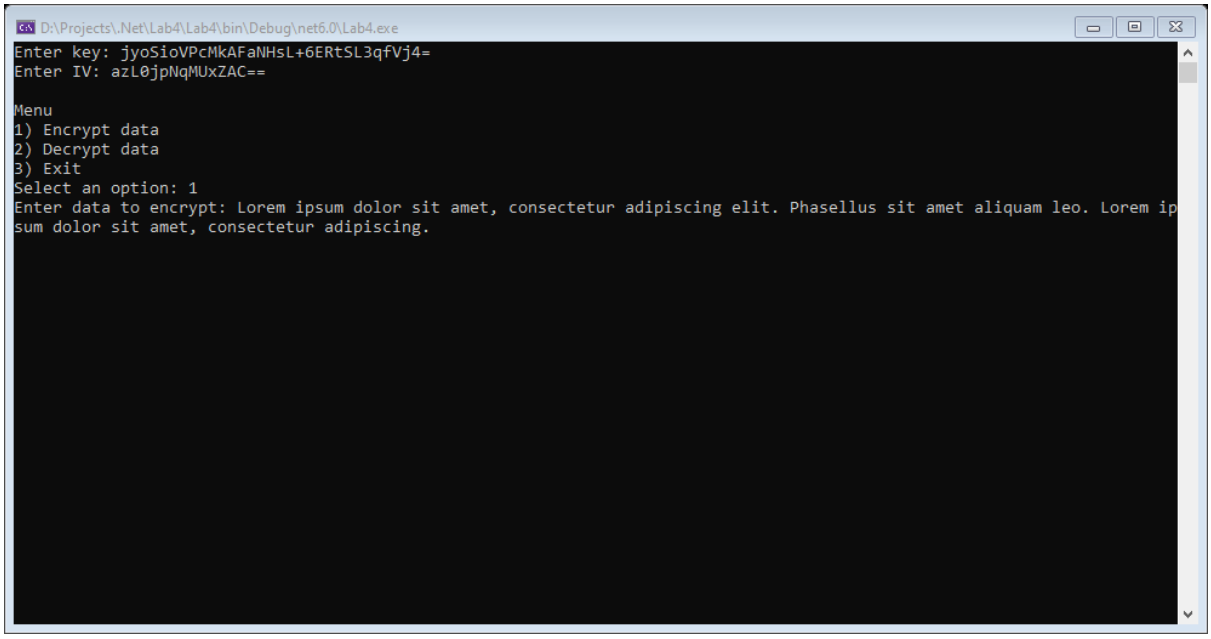


Рисунок 1.3 – обирання першої опції для шифрування даних та введення даних.

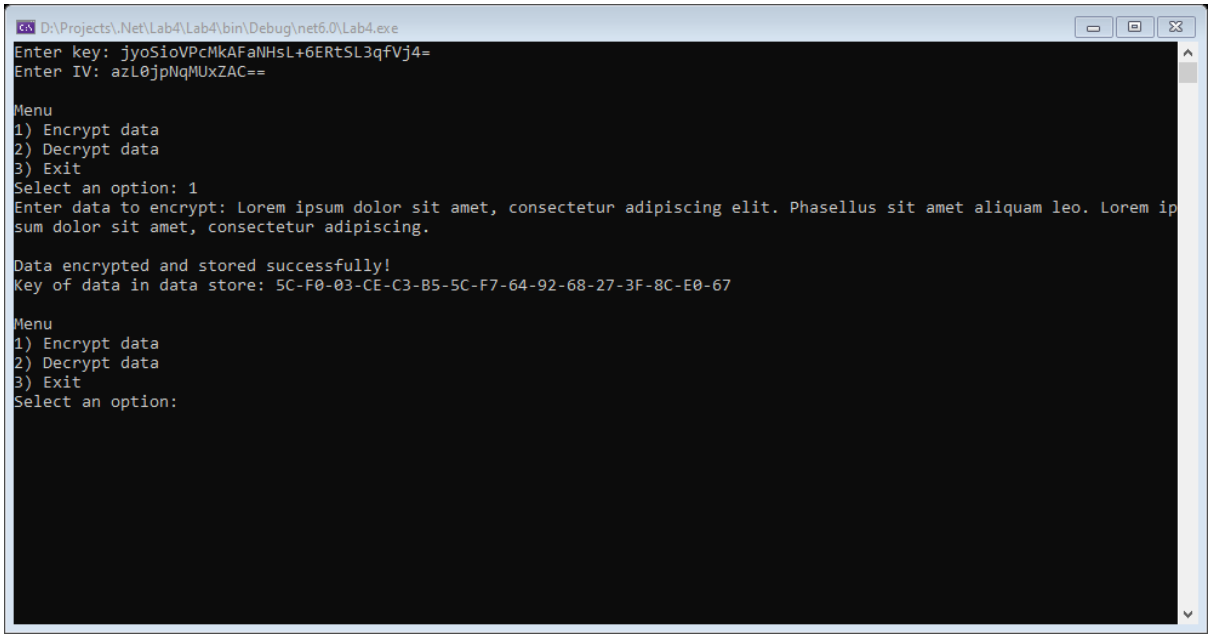


Рисунок 1.4 – результат шифрування даних, після якого повертається ключ, по якому зашифровані данні зберігаються в сховищі даних.

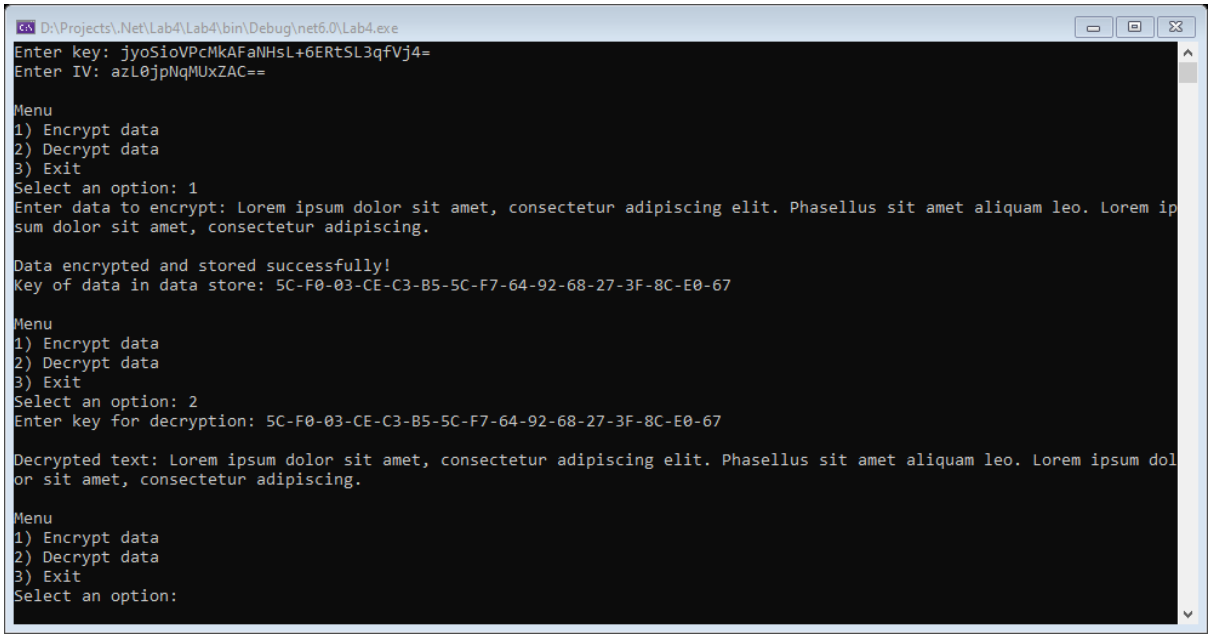


Рисунок 1.5 – обирання 2 опції для дешифрування даних, для цього вводимо ключ, за яким зашифровані данні зберігаються у сховищі даних та на вихід отримуємо дешифровані дані.

Висновок

Протягом четвертого комп’ютерного практикуму ми ознайомитися зі структурними шаблонами. Був розроблений консольний застосунок для надання послуг користувачу для шифрування та дешифрування даних. Під час розробки були застосовані такі структурні паттерни, як фасад, адаптер, та замісник(Proxy). Було реалізовано два проекти: бібліотеку класів для сервісів криптографії, яка виконує роль стороннього сервісу, та проект з сховищем даних і адаптером інтерфейсу зовнішньої системи, який виконує роль внутрішньої системи.