**Міністерство освіти і науки України**

**Тернопільський національний технічний університет ім. І. Пулюя**

Кафедра комп’ютерних наук

**ЗВІТ**

про виконання лабораторної роботи №4

з дисципліни «Технологія створення програмних продуктів»

на тему: «Проєктування програмного забезпечення з використанням патернів»

Варіант – 7

**Виконала:**

Студент СТ-21

Кальві Е.

**Прийняв:**

Палка О. В.

**Тернопіль 2025**

**Тема роботи:** Проєктування програмного забезпечення з використанням патернів.

**Мета роботи:** Ознайомитися з основними патернами проєктування та навчитися їх використовувати для створення ефективних і підтримуваних програмних рішень.

**Завдання на лабораторну роботу**

|  |  |
| --- | --- |
| **Варіант** | **Предметні області та патерни ПЗ** |
| **7** | Singleton → Менеджер підключень до бази даних Створіть DatabaseConnection, який гарантує використання лише одного екземпляра підключення.  Adapter → Робота з API різних соцмереж Реалізуйте SocialMediaAdapter, який забезпечує уніфікований доступ до API Facebook, Twitter, Instagram.   Observer → Система сповіщень у біржовому застосунку Користувач може підписатися на зміну вартості акцій та отримувати миттєві сповіщення. |

**Мета проєкту**

Метою даного проєкту є реалізація програмного забезпечення, у якому демонструється використання трьох шаблонів проєктування:

* **Singleton** — для створення менеджера підключень до бази даних, що гарантує існування лише одного екземпляра з'єднання;
* **Adapter** — для забезпечення уніфікованого доступу до API різних соціальних мереж (Facebook, Twitter, Instagram);
* **Observer** — для реалізації системи сповіщень у біржовому застосунку, де користувачі можуть підписуватись на зміну вартості акцій та отримувати миттєві повідомлення.

Лістинг 1 – Програма

using System;

using System.Collections.Generic;

namespace DesignPatternsExample

{

// Singleton → Менеджер підключень до бази даних

public class DatabaseConnection

{

private static DatabaseConnection? \_instance;

private static readonly object \_lock = new object();

private DatabaseConnection()

{

Console.WriteLine("Database connection established.");

}

public static DatabaseConnection Instance

{

get

{

lock (\_lock)

{

return \_instance ??= new DatabaseConnection();

}

}

}

public void ExecuteQuery(string query)

{

Console.WriteLine($"Executing query: {query}");

}

}

// Adapter → Робота з API різних соцмереж

public interface ISocialMediaAdapter

{

void PostMessage(string message);

}

public class FacebookAPI

{

public void SendFacebookPost(string text)

{

Console.WriteLine($"[Facebook] Posted: {text}");

}

}

public class TwitterAPI

{

public void Tweet(string content)

{

Console.WriteLine($"[Twitter] Tweeted: {content}");

}

}

public class InstagramAPI

{

public void ShareStory(string message)

{

Console.WriteLine($"[Instagram] Story shared: {message}");

}

}

public class SocialMediaAdapter : ISocialMediaAdapter

{

private readonly FacebookAPI \_facebook = new();

private readonly TwitterAPI \_twitter = new();

private readonly InstagramAPI \_instagram = new();

public void PostMessage(string message)

{

\_facebook.SendFacebookPost(message);

\_twitter.Tweet(message);

\_instagram.ShareStory(message);

}

}

// Observer → Система сповіщень у біржовому застосунку

public interface IStockObserver

{

void Update(string stockSymbol, double price);

}

public class StockObserver : IStockObserver

{

private readonly string \_name;

public StockObserver(string name)

{

\_name = name;

}

public void Update(string stockSymbol, double price)

{

Console.WriteLine($"{\_name} received update: {stockSymbol} is now {price} USD");

}

}

public class Stock

{

private readonly List<IStockObserver> \_observers = new();

private readonly string \_symbol;

private double \_price;

public Stock(string symbol)

{

\_symbol = symbol;

}

public void Subscribe(IStockObserver observer)

{

\_observers.Add(observer);

}

public void Unsubscribe(IStockObserver observer)

{

\_observers.Remove(observer);

}

public void SetPrice(double price)

{

\_price = price;

NotifyObservers();

}

private void NotifyObservers()

{

foreach (var observer in \_observers)

{

observer.Update(\_symbol, \_price);

}

}

}

// Демонстрація

class Program

{

static void Main(string[] args)

{

Console.WriteLine("=== Singleton: Database Connection ===");

var db = DatabaseConnection.Instance;

db.ExecuteQuery("SELECT \* FROM Users");

Console.WriteLine("\n=== Adapter: Social Media Posting ===");

ISocialMediaAdapter adapter = new SocialMediaAdapter();

adapter.PostMessage("New feature released!");

Console.WriteLine("\n=== Observer: Stock Price Notification ===");

var stock = new Stock("AAPL");

var user1 = new StockObserver("Alice");

var user2 = new StockObserver("Bob");

stock.Subscribe(user1);

stock.Subscribe(user2);

stock.SetPrice(155.30);

stock.SetPrice(160.75);

Console.WriteLine("\nApplication finished. Press any key to exit...");

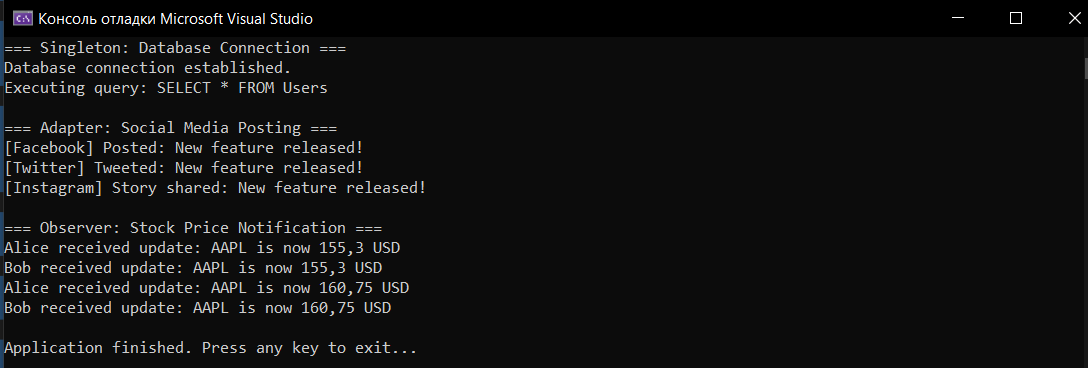
Console.ReadKey();

}

}

}

На рисунку 1 зображено результат виконання програми. У цій програмі реалізовано використання патернів: **Singleton**, який реалізує менеджер підключень до бази даних та гарантує використання лише одного екземпляра з'єднання; **Adapter**, який забезпечує уніфікований доступ до API соціальних мереж (Facebook, Twitter, Instagram); **Observer**, який реалізує систему сповіщень у біржовому застосунку, де користувач отримує повідомлення про зміну вартості акцій.

Рисунок 1 – Результат виконання програми з використання патернів

**Висновок:** виконавши цю лабораторну роботу, було ознайомлено з основними патернами проєктування та вивчино їх використовувати для створення ефективних і підтримуваних програмних рішень.