Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7

з дисципліни <<Технології розробки програмного забезпечення>>

Тема <<ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»>>

Виконав:
студент ІА-23
Содолиснький Вадим

Перевірив:
Мягкий М. Ю.

Київ 2024

Тема: ШАБЛОН «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»

Хід роботи

Варіант №13

Office communicator

Мережевий комунікатор для офісу повинен нагадувати функціонал програми Skype з можливостями голосового / відео / конференц-зв'язку, відправки текстових повідомлень і файлів (можливо, оффлайн), веденням організованого списку груп / контактів.

Короткий опис патернів

Патерн Mediator

Mediator — це шаблон, який зменшує залежності між об'єктами, виступаючи посередником для їхньої комунікації. Замість того щоб об'єкти взаємодіяли безпосередньо, вони надсилають запити до посередника, який координує їхню роботу. Це дозволяє уникнути сильного зв’язування між об'єктами, покращує читабельність і спрощує підтримку системи. Mediator часто використовується в системах з великою кількістю об'єктів, які взаємодіють між собою.

Патерн Facade

Facade — це шаблон, який надає спрощений інтерфейс для роботи зі складною підсистемою. Він приховує деталі реалізації і забезпечує легкий доступ до основного функціоналу. Фасад допомагає знизити залежності між клієнтським кодом і складною підсистемою, що робить систему більш зрозумілою і легкою для використання. Наприклад, у системах керування мультимедіа фасад може об'єднувати функції для роботи з аудіо- та відеофайлами, приховуючи складні API викликів.

Патерн Bridge

Bridge — це шаблон, який розділяє абстракцію та реалізацію, дозволяючи їм розвиватися незалежно. Він створює міст між абстракцією і її конкретними реалізаціями, забезпечуючи гнучкість у розширенні. Шаблон корисний у системах, де потрібно незалежно змінювати абстрактну логіку й конкретні деталі реалізації.

Патерн Template Method

Template Method — це шаблон, який визначає загальну структуру алгоритму в базовому класі, делегуючи реалізацію деяких кроків підкласам. Це дозволяє змінювати частини алгоритму, не порушуючи його структуру. Template Method часто використовується для побудови фреймворків, де основна логіка незмінна, а конкретні деталі можуть бути реалізовані розробниками.

Використання патерну Bridge

Переваги

Патерн дозволяє відокремити високорівневу логіку (абстракцію) від низькорівневої реалізації, що дає змогу незалежно змінювати їх без впливу один на одного.

Bridge дозволяє уникнути експоненційного зростання кількості класів. Наприклад, замість створення окремих класів для кожної комбінації платформ і функціональностей, вони розділяються на дві ієрархії.

Проблема

Результат відповіді від сервера - це об'єкти різних класів і щоб їх обробити потрібно окремі класи, що викликає велику кількість класів, які виконують одну функцію.

Рішення

Створено один клас ServerResponse, який містить поле з об'єктом відповіді. Таким чином замість створення великої кількість класів

створюється один, який зберігає об'єкт в залежності від відповіді сервера.

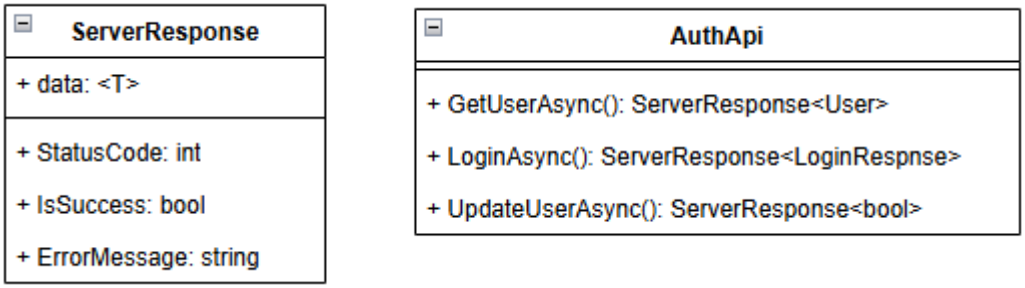

Рис. 1 – Реалізація патерну Bridge

```
public class ServerResponse<T>
{
    59 references
    public T? Data { get; set; }
    6 references
    public int StatusCode { get; set; }
    86 references
    public bool IsSuccess { get; set; }
    10 references
    public string? ErrorMessage { get; set; }

    19 references
    public ServerResponse(HttpResponseMessage response)
    {
        if(Data is not bool) Data = response.Content.ReadFromJsonAsync<T>().Result;
        StatusCode = ((int)response.StatusCode);
        IsSuccess = response.IsSuccessStatusCode;
        ErrorMessage = response.ReasonPhrase;
    }

    22 references
    public ServerResponse(T? data, int statusCode, bool isSuccess, string? errorMessage)
    {
        Data = data;
        StatusCode = statusCode;
        IsSuccess = isSuccess;
        ErrorMessage = errorMessage;
    }
}
```

Рис. 2 – Клас ServerResponse

Клас ServerResponse використовується для спрощення обробки відповідей сервера у клієнтських додатках. Він є узагальненим (generic), що дозволяє працювати з різними типами даних у відповіді сервера. Клас містить властивості для зберігання отриманих даних,

коду статусу HTTP, індикатора успішності виконання запиту та повідомлення про помилку.

Клас має два конструктори. Перший конструктор приймає об'єкт HttpResponseMessage і автоматично обробляє його: десеріалізує вміст у заданий тип, зберігає код статусу, визначає успішність запиту та, у разі помилки, записує її текст. Другий конструктор дозволяє вручну створити об'єкт ServerResponse, передаючи дані, код статусу, індикатор успіху та повідомлення про помилку. Це зручно для обробки виняткових ситуацій, коли HTTP-запит не виконується.

```
public async Task<ServerResponse<User>> GetUserAsync(string token)
{
    _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
    try
    {
        var response = await _httpClient.GetAsync(_url + "/get");
        return new ServerResponse<User>(response);
    }
    catch (Exception e)
    {
        return new ServerResponse<User>(null, 500, false, e.Message);
    }
}
```

Рис. 3 – Застосувавання класу ServerResponse для обробки відповіді сервера

Метод GetUserAsync() демонструє, як використовується клас ServerResponse для обробки відповіді сервера. У цьому методі виконується HTTP-запит для отримання даних про користувача. У разі успішного виконання запиту дані десеріалізуються і передаються в екземпляр ServerResponse, який повертається користувачеві. Якщо під час виконання запиту виникає виняток, створюється об'єкт ServerResponse із кодом статусу 500 та текстом помилки.

Висновок: я використав патерн Bridge для уніфікації обробки відповіді від сервера. Створивши клас ServerResponse я централізував обробку відповідей сервера, уніфікував логіку роботи з HTTP-запитами та спростив обробку помилок. Завдяки використанню узагальнень, клас ServerResponse є універсальним і підходить для

роботи з різними типами даних, що підвищує читабельність і підтримуваність коду. Таким чином реалізація відповідей від сервера та відображення контенту на стороні клієнта не залежать одна від одної.