Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Технології розроблення програмного забезпечення

Лабораторна робота №7

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

Виконала
студентка групи ІА–24:
Кармазіна А. В.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

Завдання

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

Тема: 26 Download manager (iterator, command, observer, template method, composite, p2p)

Інструмент для скачування файлів з інтернету по протоколах http або https з можливістю продовження завантаження в зупиненому місці, розподілу швидкостей активним завантаженням, ведення статистики завантажень, інтеграції в основні браузери (firefox, opera, internet explorer, chrome) (https://github.com/Karmazinanastya/TRPZ_Karmazina_labs/tree/main/FileDownloader)

Теоретичні відомості

1. Mediator (Посередник)

Mediator — це шаблон проектування, який визначає об'єкт, що інкапсулює спосіб взаємодії між колекцією об'єктів. У цьому шаблоні об'єкти не взаємодіють безпосередньо між собою, а через посередника. Це дозволяє зменшити залежності між класами і сприяє слабкій зв'язності.

Основні компоненти:

Mediator: інтерфейс для посередника, який містить методи для спілкування між об'єктами.

ConcreteMediator: реалізація посередника, яка координує взаємодії між конкретними об'єктами.

Colleagues: об'єкти, які взаємодіють через посередника, але не безпосередньо один з одним.

2. Facade (Фасад)

Facade — це шаблон проектування, який надає спрощений інтерфейс для складної підсистеми. Замість того, щоб взаємодіяти з багатьма складними об'єктами, користувач взаємодіє лише з одним фасадом, який делегує запити до відповідних об'єктів підсистеми.

Основні компоненти:

Facade: об'єкт, який надає спрощений інтерфейс для складних підсистем.

Subsystem classes: класи, які реалізують складну бізнес-логіку, але користувач взаємодіє лише з фасадом.

3. Bridge (Міст)

Bridge — це шаблон проектування, який дозволяє розділити абстракцію і її реалізацію таким чином, щоб вони могли змінюватися незалежно один від одного. Шаблон дозволяє зберігати гнучкість і масштабованість, оскільки нові абстракції та реалізації можуть бути додані без змін в існуючому коді.

Основні компоненти:

Abstraction: абстракція, яка делегує завдання до реалізації.

RefinedAbstraction: розширена версія абстракції.

Implementor: інтерфейс реалізації, який визначає загальні методи.

ConcreteImplementor: конкретна реалізація інтерфейсу реалізації.

4. Template Method (Шаблонний метод)

Template Method — це шаблон проектування, який визначає скелет алгоритму в методі, залишаючи деякі кроки реалізації підкласам. Це дозволяє підкласам змінювати частини алгоритму без зміни його структури.

Основні компоненти:

AbstractClass: клас, що визначає шаблонний метод, який містить загальні кроки алгоритму.

ConcreteClass: підклас, який реалізує деякі кроки алгоритму.

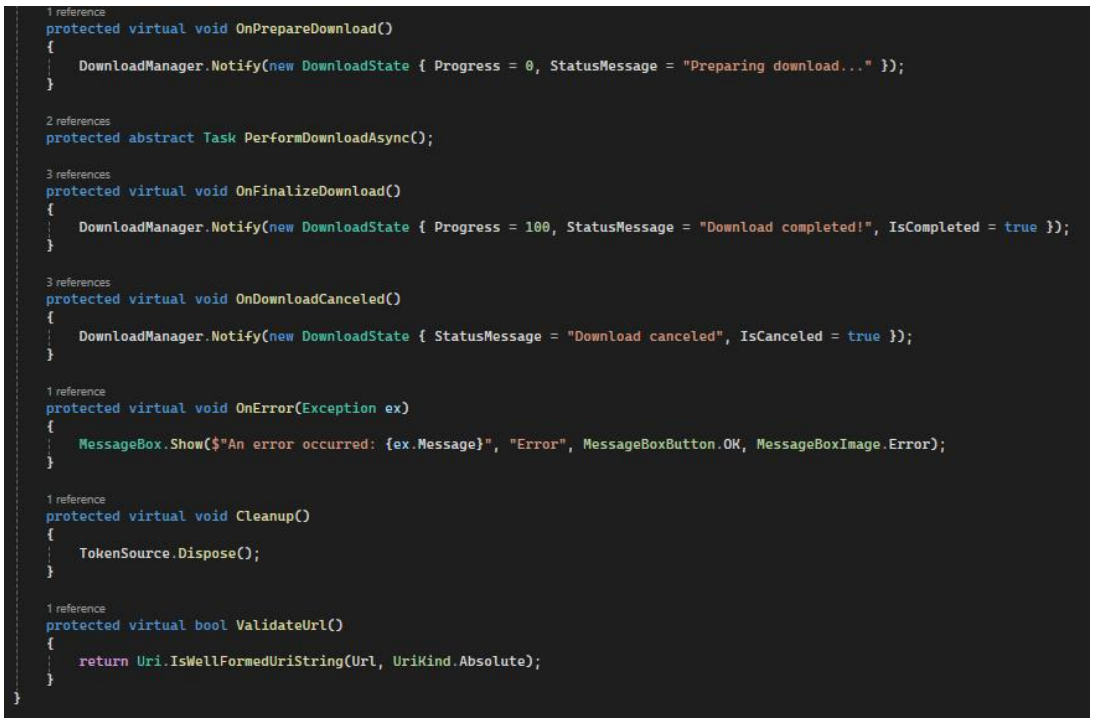
Хід роботи

Реалізація шаблону **Template Method**

Шаблон Template Method є частиною шаблонів поведінки і дозволяє визначити загальну структуру алгоритму в методі, залишаючи деякі кроки реалізації підкласам. Це дозволяє підкласам змінювати частини алгоритму, не змінюючи загальну структуру.

Реалізовано шаблон Template Method для завантаження файлів через HTTP. Клас BaseDownloader визначає загальний алгоритм завантаження, а клас HttpFileDownloader реалізує конкретні кроки завантаження.

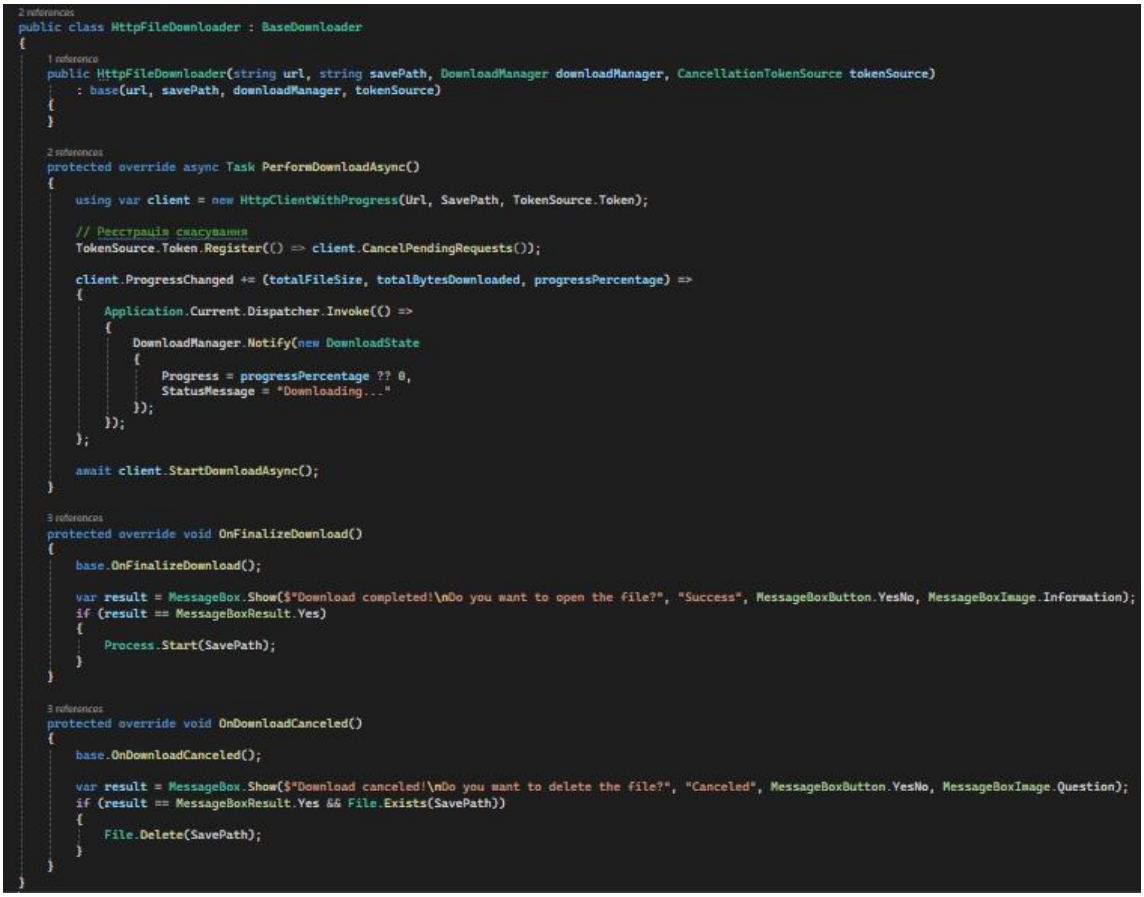
```
public abstract class BaseDownloader
{
    3 references
    protected string Url { get; }
    5 references
    protected string SavePath { get; }
    5 references
    protected DownloadManager DownloadManager { get; }
    4 references
    protected CancellationTokenSource TokenSource { get; }

    1 reference
    protected BaseDownloader(string url, string savePath, DownloadManager downloadManager, CancellationTokenSource tokenSource)
    {
        Url = url;
        SavePath = savePath;
        DownloadManager = downloadManager;
        TokenSource = tokenSource;
    }

    1 reference
    public async Task DownloadFileAsync()
    {
        if (!ValidateUrl())
        {
            MessageBox.Show("Invalid URI, please enter a valid one.");
            return;
        }

        try
        {
            OnPrepareDownload();
            await PerformDownloadAsync();
            OnFinalizeDownload();
        }
        catch (TaskCanceledException)
        {
            OnDownloadCanceled();
        }
        catch (Exception ex)
        {
            OnError(ex);
        }
        finally
        {
            Cleanup();
        }
    }
```



```
    1 reference
    protected virtual void OnPrepareDownload()
    {
        DownloadManager.Notify(new DownloadState { Progress = 0, StatusMessage = "Preparing download..." });
    }

    2 references
    protected abstract Task PerformDownloadAsync();

    3 references
    protected virtual void OnFinalizeDownload()
    {
        DownloadManager.Notify(new DownloadState { Progress = 100, StatusMessage = "Download completed!", IsCompleted = true });
    }

    3 references
    protected virtual void OnDownloadCanceled()
    {
        DownloadManager.Notify(new DownloadState { StatusMessage = "Download canceled", IsCanceled = true });
    }

    1 reference
    protected virtual void OnError(Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
    }

    1 reference
    protected virtual void Cleanup()
    {
        TokenSource.Dispose();
    }

    1 reference
    protected virtual bool ValidateUrl()
    {
        return Uri.IsWellFormedUriString(Url, UriKind.Absolute);
    }
}
```

Рис. 1 - BaseDownloader

BaseDownloader визначає шаблонний метод DownloadFileAsync, що керує процесом завантаження. Цей клас містить загальну логіку для роботи з усіма видами завантажень, а конкретні дії реалізуються в підкласах.



```
2 references
public class HttpFileDownloader : BaseDownloader
{
    1 reference
    public HttpFileDownloader(string url, string savePath, DownloadManager downloadManager, CancellationTokenSource tokenSource)
        : base(url, savePath, downloadManager, tokenSource)
    {
    }

    2 references
    protected override async Task PerformDownloadAsync()
    {
        using var client = new HttpClientWithProgress(Url, SavePath, TokenSource.Token);

        // Реєстрація скасування
        TokenSource.Token.Register(() => client.CancelPendingRequests());

        client.ProgressChanged += (totalFileSize, totalBytesDownloaded, progressPercentage) =>
        {
            Application.Current.Dispatcher.Invoke(() =>
            {
                DownloadManager.Notify(new DownloadState
                {
                    Progress = progressPercentage ?? 0,
                    StatusMessage = "Downloading..."
                });
            });
        };

        await client.StartDownloadAsync();
    }

    3 references
    protected override void OnFinalizeDownload()
    {
        base.OnFinalizeDownload();

        var result = MessageBox.Show($"Download completed!\nDo you want to open the file?", "Success", MessageBoxButton.YesNo, MessageBoxImage.Information);
        if (result == MessageBoxResult.Yes)
        {
            Process.Start(SavePath);
        }
    }

    3 references
    protected override void OnDownloadCanceled()
    {
        base.OnDownloadCanceled();

        var result = MessageBox.Show($"Download canceled!\nDo you want to delete the file?", "Canceled", MessageBoxButton.YesNo, MessageBoxImage.Question);
        if (result == MessageBoxResult.Yes && File.Exists(SavePath))
        {
            File.Delete(SavePath);
        }
    }
}
```

Рис. 2 - Клас HttpFileDownloader

HttpFileDownloader є конкретною реалізацією класу BaseDownloader. У ньому реалізовано метод PerformDownloadAsync, який виконує завантаження файлу за допомогою класу HttpClientWithProgress. Інші кроки (підготовка до завантаження, фіналізація, обробка скасування та помилок) можуть бути змінені або залишені за замовчуванням.

HttpFileDownloader наслідує BaseDownloader та реалізує абстрактний метод PerformDownloadAsync(). В ньому виконується конкретне завантаження файлу з використанням HttpClientWithProgress.

Методи OnFinalizeDownload() та OnDownloadCanceled() реалізують додаткові дії після завершення або скасування завантаження (відкриття файлу або видалення).

Висновок

Шаблон Template Method є потужним інструментом для визначення загальної структури алгоритму в базовому класі, при цьому залишаючи певні етапи алгоритму для реалізації в підкласах. Це дозволяє спростити код, зробивши його більш гнучким і підтримуваним. У реалізації шаблонного методу для завантаження файлів, BaseDownloader задає загальну логіку завантаження, в той час як HttpFileDownloader реалізує специфічні деталі завантаження, такі як прогрес і обробка помилок. Такий підхід дозволяє легко додавати нові типи завантаження без змін у основному коді, підвищуючи масштабованість і підтримуваність системи.