Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «ШАБЛОН «MEDIATOR», «FACADE»,
«BRIDGE», «TEMPLATE METHOD»»
Варіант №26

Виконав:
студент групи ІА-23
Мозоль В.О

Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Тема.
Шаблони «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»

Мета.
Метою лабораторної роботи є поглиблене вивчення та практичне опанування шаблонів проєктування, зокрема шаблонів «Mediator», «Facade», «Bridge» та «Template Method». Реалізація шаблону Template Method згідно обраної теми. Формування навичок застосування об'єктно-орієнтованого підходу при розробці складних програмних системВдосконалення вмінь проєктування архітектури програмного забезпечення з використанням сучасних патернів

Завдання.
1 . Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми

Обрана 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).

Короткі теоретичні відомості.

Основні принципи проєктування програмного забезпечення

1. Принцип DRY (Don't Repeat Yourself)

Цей принцип наголошує на необхідності уникнення повторень у вихідному коді.

Основні причини дотримання принципу:

- Зменшення обсягу коду
- Полегшення читабельності
- Спрощення підтримки та модифікації
- Мінімізація ризику поширення помилок через копіювання коду

2. Принцип KISS (Keep it Simple, Stupid!)

Філософія простоти в проєктуванні систем:

- Перевага простих компонентів над складними
- Кожен компонент повинен виконувати чітко визначену функцію
- Простота сприяє надійності та зрозумілості коду

3. Принцип YOLO (You Only Load It Once!)

Оптимізація ініціалізації та конфігурації:

- Одноразове завантаження конфігураційних змінних
- Попередження проблем із продуктивністю
- Мінімізація повторних операцій введення-виведення

4. Принцип Парето (80/20)

Статистичний підхід до оптимізації:

- 80% навантаження створюється 20% компонентів
- 80% коду пишеться за 20% часу
- 80% помилок можна усунути, виправивши 20% вад

5. Принцип YAGNI (You Ain't Gonna Need It)

Уникнення зайвої складності:

- Реалізація лише необхідного функціоналу
- Відмова від передчасного узагальнення
- Фокус на актуальних вимогах

Шаблони проєктування

1. Шаблон Mediator (Посередник)

Призначення: Централізація взаємодії між компонентами через проміжний об'єкт.

Основні характеристики:

- Зменшення прямих залежностей між компонентами
- Спрощення комунікації
- Підвищення гнучкості системи

Приклад: Диспетчер керування повітряним рухом, який координує взаємодію літаків.

2. Шаблон Facade (Фасад)

Призначення: Створення уніфікованого інтерфейсу для складної підсистеми.

Основні характеристики:

- Приховування внутрішніх деталей реалізації

Приклад: Співробітник служби підтримки магазину як єдина точка взаємодії для клієнта.

3. Шаблон Bridge (Міст)

Призначення: Роз'єднання абстракції та реалізації.

Основні характеристики:

- Зменшення кількості класів при комбінуванні властивостей
- Незалежний розвиток ієрархій абстракції та реалізації
- Гнучкість додавання нових абстракцій та реалізацій

Приклад: Незалежний розвиток класів фігур та кольорів.

4. Шаблон Template Method (Шаблонний метод)

Призначення: Визначення кістяка алгоритму з можливістю зміни окремих кроків.

Основні характеристики:

- Винесення загальної логіки в базовий клас
- Можливість перевизначення окремих кроків у підкласах

Приклад: Стандартизований процес будівництва будинків з можливістю локальних варіацій.

Хід роботи.

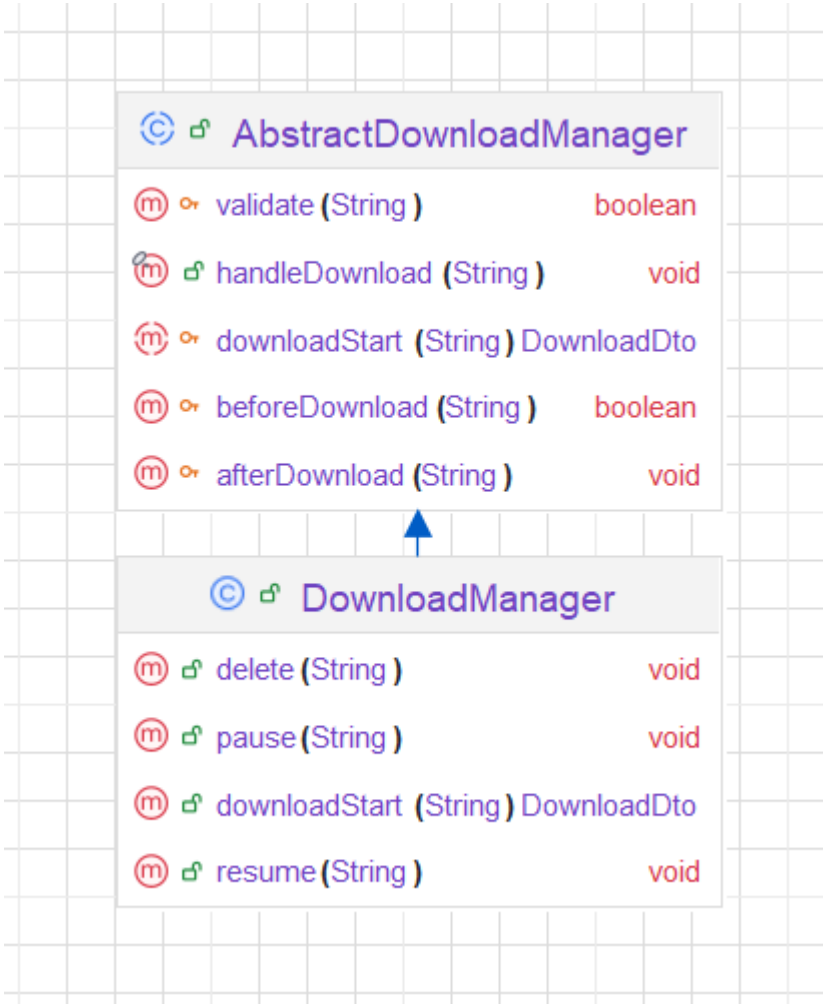

Рисунок №1 – Діаграма класів , згенерована IDE, реалізації шаблону Observer

Template Method — це поведінковий паттерн проєктування, який визначає структуру алгоритму в базовому класі та дозволяє підкласам змінювати деякі його кроки, не змінюючи загальної логіки.

Основні компоненти:

1. Абстрактний клас (AbstractDownloadManager):
    - Містить шаблонний метод, що визначає основні етапи алгоритму.
    - Визначає методи, що можуть бути абстрактними або з базовою реалізацією.
2. Конкретний клас (DownloadManager):
    - Реалізує або перевизначає окремі методи, щоб уточнити поведінку на певних етапах.

Приклад із UML-діаграми:

1. Шаблонний метод:
    - downloadStart(String) : DownloadDto — міститься в базовому класі AbstractDownloadManager і викликає такі методи:
        - beforeDownload(String) — перевірка перед завантаженням.
        - handleDownload(String) — основна логіка завантаження.
        - afterDownload(String) — дії після завершення.
2. Змінні кроки алгоритму:
    - У DownloadManager реалізуються методи для таких дій, як delete(String), pause(String), resume(String).

Код реалізації паттерну можна переглянути у GitHub репозиторії у папці DownloadManager або у Додатку А.

Робота паттерну.

Для демонстрації роботи паттерну будемо симулювати завантаження файлу через шаблонний метод. (Код для тестування є у GitHub репозиторії або у Додатку А)

Результат виконання коду:

```
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
Validate url: https://sabnzbd.org/tests/internetspeed/50MB.bin
Url validate:https://sabnzbd.org/tests/internetspeed/50MB.bin
Preparing to download: https://sabnzbd.org/tests/internetspeed/50MB.bin
Start download from abstract template method
Download started successfully for: https://sabnzbd.org/tests/internetspeed/50MB.bin
Download started
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: DOWNLOADING
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: DOWNLOADING
```

Рисунок №2 – Виконання методів по порядку відповідно до Template Method

```
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: DOWNLOADING
GUI -> Downloaded: 11,82%, Speed: 10386,39 KB/s, Remaining: 4 sec
STATISTICS -> DOWNLOADING download: https://sabnzbd.org/tests/internetspeed/50MB.bin
Download paused
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: PAUSED
GUI -> Downloaded: 11,82%, Speed: 10386,39 KB/s, Remaining: 4 sec
STATISTICS -> PAUSED download: https://sabnzbd.org/tests/internetspeed/50MB.bin
Resuming download: https://sabnzbd.org/tests/internetspeed/50MB.bin
Download resumed
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: DOWNLOADING
GUI -> Downloaded: 11,82%, Speed: 10386,39 KB/s, Remaining: 4 sec
STATISTICS -> DOWNLOADING download: https://sabnzbd.org/tests/internetspeed/50MB.bin
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: DOWNLOADING
GUI -> Downloaded: 11,82%, Speed: Infinity KB/s, Remaining: 0 sec
```

Рисунок №3 – Зупинка та продовження завантаження

```
STATISTICS -> DOWNLOADING download: https://sabnzbd.org/tests/internetspeed/50MB.bin
Cancelling download: https://sabnzbd.org/tests/internetspeed/50MB.bin
Download deleted!
LOG -> Download https://sabnzbd.org/tests/internetspeed/50MB.bin status: CANCELLED
GUI -> Downloaded: 17,96%, Speed: 26721,56 KB/s, Remaining: 1 sec
STATISTICS -> CANCELLED download: https://sabnzbd.org/tests/internetspeed/50MB.bin
```

Рисунок №4 – Видалення завантаження

Висновки.

За результатами виконання лабораторної роботи можна зробити наступні висновки. Проведено детальне вивчення шаблонів проєктування, зокрема «Template Method» Застосовано патерн Template Method для створення гнучкого та розширюваного механізму завантаження файлів. Розвинуто вміння застосовувати шаблони проєктування у розробці програмного забезпечення.

Додаток А.

AbstractDownloadManager.java

```
public abstract class AbstractDownloadManager {

    protected final Map<String, DownloadDto> downloads = new ConcurrentHashMap<>();
    protected final ExecutorService executorService = Executors.newCachedThreadPool();

    public final void handleDownload(String url) {
        if (validate(url)) {
            if (!beforeDownload(url)) {
                return;
            }
            System.out.println("Start download from abstract template method");
            DownloadDto download = downloadStart(url);
            downloads.put(url, download);
            executorService.submit(download);
            afterDownload(url);
        }
    }

    protected boolean validate(String url) {
        System.out.println("Validate url: " + url);
        if (downloads.containsKey(url)) {
            System.out.println("Download already exists for: " + url);
            return false;
        }
        System.out.println("Url validate:" + url);
        return true;
    }

    protected boolean beforeDownload(String url) {
        System.out.println("Preparing to download: " + url);
        return true;
    }

    protected abstract DownloadDto downloadStart(String url);

    protected void afterDownload(String url) {
        System.out.println("Download started successfully for: " + url);
    }
}
```

DownloadManager.java

```
public class DownloadManager extends AbstractDownloadManager {

    private final Map<String, DownloadDto> downloads = new ConcurrentHashMap<>();
    private final ExecutorService executorService = Executors.newCachedThreadPool();

    public void resume(String url) {
        DownloadDto download = downloads.get(url);
        if (download == null) {
            System.out.println("No download found for: " + url);
            return;
        }

        if (download.getStatus() != DownloadStatus.PAUSED) {
            System.out.println("Download is not paused for: " + url);
            return;
        }

        System.out.println("Resuming download: " + url);
        download.resume();
        executorService.submit(download);
    }

    public void pause(String url) {
        DownloadDto download = downloads.get(url);
        if (download == null) {
            System.out.println("No download found for: " + url);
```

```
            return;
        }

        if (download.getStatus() != DownloadStatus.DOWNLOADING) {
            System.out.println("Download is not active for: " + url);
            return;
        }

        System.out.println("Pausing download: " + url);
        download.pause();

    }

    public void delete(String url) {
        DownloadDto download = downloads.get(url);
        if (download == null) {
            System.out.println("No download to cancel for: " + url);
            return;
        }

        System.out.println("Cancelling download: " + url);
        download.setStatus(DownloadStatus.CANCELLED);
        downloads.remove(url);
    }

    public DownloadDto downloadStart(String url) {

        System.out.println("Starting download: " + url);
        DownloadDto download = new DownloadDto(url);

        download.attach(new LogObserver());
        download.attach(new GUIObserver());
        download.attach(new StatisticObserver());

        downloads.put(url, download);
        executorService.submit(download);

        return download;
    }
```