Міністерство освіти і науки України
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 розроблення програмного забезпечення»
Тема: **«ШАБЛОНИ «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»»**
Flexible Automatical Tool

Виконав:
Студент групи ІА-22
Сидорін Д.О.

Перевірив:
Мягкий М. Ю.

Київ-2024

Зміст

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

Мета:
Ознайомитися з основними шаблонами проєктування, такими як «Mediator», «Facade», «Bridge», «Template Method», дослідити їхні принципи роботи та навчитися використовувати їх для створення гнучкого та масштабованого програмного забезпечення.

Завдання:

Інструмент автоматизації (стратегія, прототип, абстрактна фабрика, міст, композит, SOA)
Десктопний додаток для автоматизації повсякденних завдань із можливістю створення правил (аналогічно сервісу IFTTT), запису макросів (натискання клавіш, дії миші) та використання планувальника завдань. Додаток забезпечує функції, як-от автоматичне завантаження нових серій серіалів, книг чи інших файлів у визначений час, зміну статусів у месенджерах (наприклад, встановлення статусу "відсутній" у Skype при довгій неактивності), а також виконання завдань за розкладом (наприклад, запуск роздачі торрентів о 5 ранку).

Хід роботи

1. Реалізувати не менше 3-х класів відповідно до обраної теми

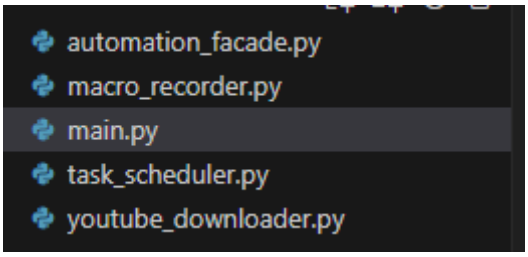

Рис. 1 — Структура проекту

1. YouTubeDownloader

Призначення: Клас відповідає за завантаження відео з YouTube-каналів.

Методи:

- download_video(channel_url, video_format="mp4"):
  - Завантажує відео з вказаного каналу.
  - Параметри:
    - channel_url (str): URL каналу або відео.
    - video_format (str, за замовчуванням "mp4"): Формат завантажуваного відео.

    - Повертає: назву завантаженого відеофайлу у вигляді рядка ("video.{video_format}").
    - **Приклад**: "video.mp4".

---

2. MacroRecorder

Призначення: Клас для запису та виконання макросів, що імітують дії користувача (натискання клавіш, кліки миші тощо).

Методи:

- record_macro(macro_name):
    - Записує макрос із заданою назвою.
    - Параметри:
        - macro_name (str): Назва макросу.
    - Повертає: назву файлу, у якому збережений макрос ("{macro_name}.macro").
- execute_macro(macro_file):
    - Виконує записаний макрос.
    - Параметри:
        - macro_file (str): Назва файлу макросу.

---

3. TaskScheduler

Призначення: Клас для планування завдань, які будуть виконані в майбутньому.

Методи:

- schedule_task(task_name, time):
    - Планує виконання завдання на заданий час.
    - Параметри:
        - task_name (str): Назва завдання.
        - time (str): Час, коли завдання має бути виконане (наприклад, "5:00 AM").

---

4. AutomationFacade

Призначення: Фасадний клас, який об'єднує функціональність кількох компонентів і спрощує взаємодію з ними.

Методи:

- __init__():
    - Ініціалізує об'єкти класів YouTubeDownloader, MacroRecorder та TaskScheduler.

- download_and_schedule_video(channel_url, video_format, time):
  - Завантажує відео з YouTube і планує його роздачу.
  - Параметри:
    - channel_url (str): URL каналу YouTube.
    - video_format (str): Формат відео (наприклад, "mp4").
    - time (str): Час для роздачі (наприклад, "5:00 AM").
- record_and_execute_macro(macro_name):
  - Записує макрос і одразу виконує його.
  - Параметри:
    - macro_name (str): Назва макросу.

Як вони взаємодіють:

1. **AutomationFacade** — основна точка доступу, яка ховає складність роботи з іншими класами.
2. Клієнт викликає методи фасаду, такі як download_and_schedule_video чи record_and_execute_macro, а фасад передає виклики відповідним класам (YouTubeDownloader, MacroRecorder, TaskScheduler).
3. Наприклад, завантаження відео:
   - Клієнт викликає AutomationFacade.download_and_schedule_video().
   - Фасад використовує YouTubeDownloader для завантаження відео.
   - Потім використовує TaskScheduler для планування дій із завантаженим відео.

.

2. Реалізувати один з розглянутих шаблонів за обраною темою

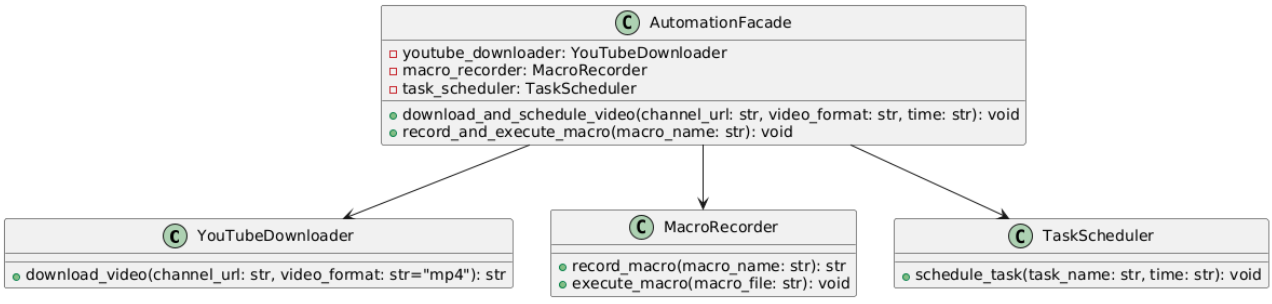

Рис. 2 — Діаграма класів

Діаграма демонструє взаємодію між компонентами патерну **Facade** для автоматизації завдань:

1. **YouTubeDownloader**, **MacroRecorder**, **TaskScheduler** — це підсистеми, які відповідають за конкретні завдання:
   - Завантаження відео.

    - Робота з макросами.
    - Планування завдань.
2. **AutomationFacade** — фасад, що виступає посередником між клієнтом та підсистемами, спрощуючи їх використання.
    - Інкапсулює складну взаємодію між підсистемами в два основні методи:
        - Завантаження відео та планування дій.
        - Запис і виконання макросів.

Зв’язки між класами показують, що фасад залежить від класів підсистем для виконання основних функцій.

Переваги використання патерну Facade

1. **Спростування інтерфейсу**:
    - Забезпечує простий і зрозумілий інтерфейс для взаємодії з комплексною системою.
    - Клієнт працює лише з фасадом, а не з окремими модулями.
2. **Інкапсуляція складності**:
    - Ховає деталі реалізації підсистем від клієнта.
    - Зміни у підсистемах не впливають на клієнтський код.
3. **Зменшення залежностей**:
    - Клієнт залежить лише від фасаду, а не від багатьох підсистем одночасно.
4. **Покращення підтримки та модульності**:
    - Кожна підсистема може розвиватися незалежно без ризику порушення роботи клієнтського коду.

Проблеми, які вирішує патерн Facade

1. **Складність системи**:
    - У складних системах із багатьма підсистемами фасад забезпечує централізовану точку доступу.
2. **Залежності між компонентами**:
    - Зменшує кількість залежностей клієнта від внутрішніх деталей реалізації.
3. **Відсутність єдиного інтерфейсу**:
    - Фасад уніфікує доступ до різних підсистем, роблячи взаємодію більш послідовною.
4. **Зміни в системі**:
    - Клієнтський код не потребує змін, якщо змінюються внутрішні деталі підсистем (оскільки фасад ховає ці деталі).

Перевірка патерну

main.py > ...

```python
from automation_facade import AutomationFacade

if __name__ == "__main__":
    # Ініціалізація фасаду
    automation_tool = AutomationFacade()

    # Тест 1: Завантаження відео та планування його роздачі
    print("=== Тест 1: Завантаження відео та планування ===")
    automation_tool.download_and_schedule_video(
        channel_url="https://youtube.com/examplechannel",
        video_format="mp4",
        time="6:00 AM"
    )

    # Тест 2: Запис та виконання макросу
    print("\n=== Тест 2: Запис та виконання макросу ===")
    automation_tool.record_and_execute_macro("TestMacro")

    print("\nПеревірка завершена. Патерн Facade працює коректно.")

```

Рис. 3 — Перевірка роботи

Процес перевірки патерну Facade

1. **Підготовка**: У файлі main.py створюється клієнтський код для взаємодії з фасадом AutomationFacade.
2. **Тест 1**:
   - Викликається метод download_and_schedule_video().
   - Фасад:
     - Завантажує відео з YouTube через YouTubeDownloader.
     - Планує його роздачу через TaskScheduler.
3. **Тест 2**:
   - Викликається метод record_and_execute_macro().
   - Фасад:
     - Записує макрос через MacroRecorder.
     - Виконує його.
4. **Запуск**:
   - Код запускається через термінал командою python main.py.
   - У консолі виводяться результати роботи методів, які демонструють взаємодію між фасадом і підсистемами.
5. **Очікуваний результат**:
   - Логічний і зрозумілий вивід у консолі підтверджує, що всі класи взаємодіють коректно, а фасад приховує складність роботи з підсистемами.

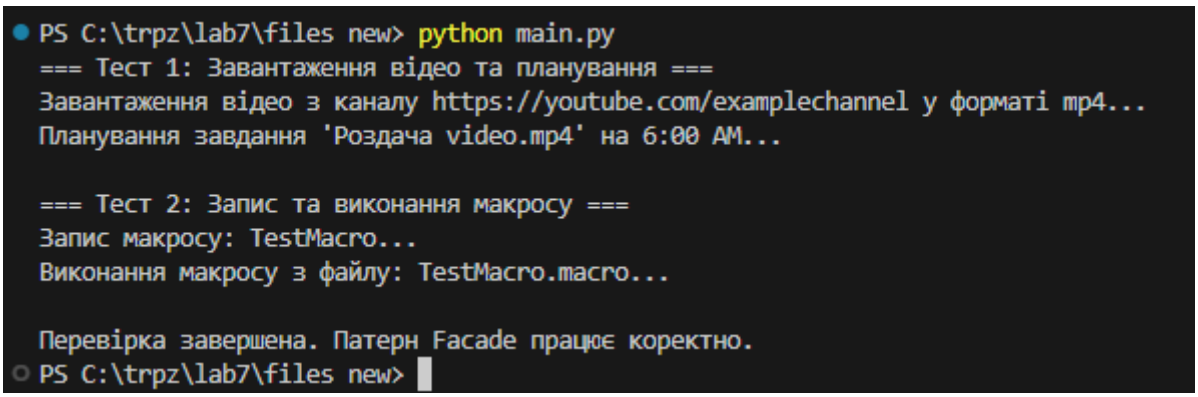

Рис. 4 — Результат роботи

- **Тест 1**:
  - Повідомлення про завантаження відео з YouTube у вибраному форматі (наприклад, mp4).
  - Повідомлення про планування роздачі завантаженого відео на заданий час.

 Тест 2:

  - Повідомлення про запис макросу з вказаною назвою.
  - Повідомлення про виконання записаного макросу.

 Фінальний висновок:

  - Повідомлення про завершення перевірки та коректну роботу патерну Facade.

Висновки:

У цій лабораторній роботі реалізувано патерн **Facade**, було досліджено структуру, призначення, переваги та недоліки, також зробив реалізацію классів, діаграмми классів, та успішне тестування паттерну.

Код: https://github.com/Lepseich/trpz/tree/main/lab7/files