Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №4

Технологія розробки програмного забезпечення

Шаблони «Singleton», «Iterator», «Proxy», «State», «Strategy»

Виконала студентка
групи ІА-23:
Шрубович Н. С.

Перевірив:
Мягкий М. Ю.

Київ 2024

Тема: Шаблони «Singleton», «Iterator», «Proxy», «State», «Strategy»

Мета: Метою даної лабораторної роботи є ознайомлення з шаблонами проєктування, зокрема з шаблоном "Strategy", та їх практичне застосування при розробці програмного забезпечення.

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

Варіант:

..3 Текстовий редактор (strategy, command, observer, template method, flyweight, SOA)

Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

ЗМІСТ

Теоретичні відомості

Шаблон проєктування (або патерни)— це формалізоване рішення типової задачі, що включає опис рішення, рекомендації щодо його застосування та уніфіковане найменування. Правильне моделювання предметної області є ключовим етапом для вибору відповідного шаблону.

Переваги використання:

- Зменшення трудовитрат і часу на проєктування;
- Надання системі гнучкості та адаптованості;
- Спрощення підтримки системи;
- Єдиний словник для комунікації розробників.

Популярні шаблони:

1. **Singleton (Одинак)** Забезпечує наявність єдиного екземпляра класу.
2. **Iterator (Ітератор)** Забезпечує доступ до елементів колекції без розкриття її внутрішньої структури.
3. **Proxy (Проксі)** Контроль доступу до об’єктів або їх заміна.
4. **State (Стан)** Зміна поведінки об'єкта залежно від його стану.
5. **Strategy (Стратегія)** Замінність алгоритмів поведінки об’єкта.

Шаблон «Стратегія» (Strategy)

Поведінковий шаблон проєктування, який дозволяє об'єкту змінювати свою поведінку під час виконання програми, обираючи одну з кількох реалізацій алгоритму (стратегії). Алгоритми інкапсулюються у вигляді окремих класів, що робить їх взаємозамінними без змін у клієнтському коді.

Основні компоненти:

1. Контекст (Context) :
   Клас, що зберігає посилання на стратегію та делегує їй виконання конкретної поведінки.
2. Стратегія (Strategy):
   Загальний інтерфейс або абстрактний клас, який визначає методи для реалізації алгоритмів.
3. Конкретні стратегії (Concrete Strategies):
   Класи, що реалізують конкретні алгоритми або поведінку.

Переваги:

- Легке переключення між алгоритмами.
- Додавання нових стратегій без змін у вже існуючому коді.
- Підвищення гнучкості та підтримуваності системи.

Недоліки:

- Зростання кількості класів у програмі.
- Відповідальність за вибір стратегії лежить на клієнтському коді.

Хід Роботи

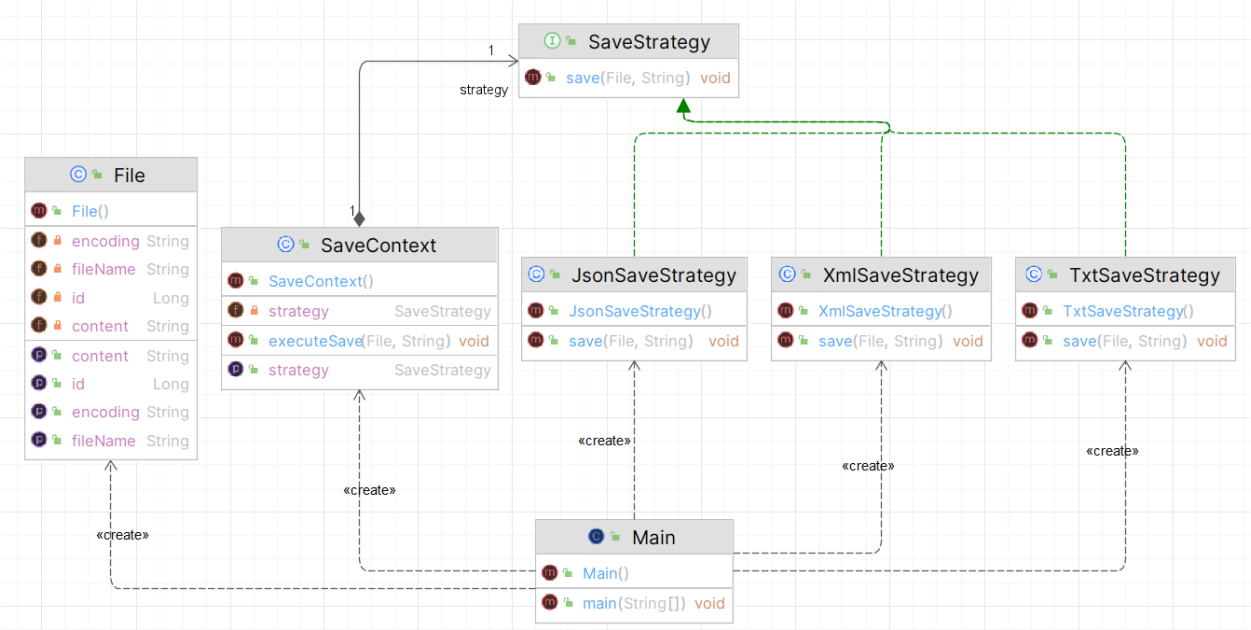

Рис 1. Діаграма класів

Діаграма класів

1. Контекст – на діаграмі SaveContext пов'язаний із загальним інтерфейсом SaveStrategy через композицію. SaveContext відповідає за вибір конкретної стратегії збереження та виклик методу save.
2. Загальний інтерфейс – інтерфейс SaveStrategy визначає метод save(File file, String path), який реалізується конкретними стратегіями. Це дозволяє обробляти файли різними способами без зміни коду контексту.
3. Конкретні стратегії – на діаграмі класів показані наступні класи:
    - JsonSaveStrategy – зберігає файл у форматі JSON.
    - XmlSaveStrategy – зберігає файл у форматі XML.
    - TxtSaveStrategy – зберігає файл у текстовому форматі .txt.

Ці стратегії реалізують метод save і забезпечують конкретний алгоритм збереження файлу.

4. Заміна алгоритмів – SaveContext може динамічно змінювати стратегію збереження, дозволяючи зберігати файл у різних форматах. Це демонструє гнучкість шаблону "Стратегія".

```
Saving file as TXT...
File saved as TXT: example
Saving file as JSON...
File saved as JSON: example
Saving file as XML...
File saved as XML: example
```

Рис 2. Застосування шаблону при реалізації програми

Робота патерну

1. Ініціалізація об'єкта файлу – створюється об'єкт класу File з відповідними властивостями: fileName, encoding, та content.
2. Створення контексту та вибір стратегії – об'єкт SaveContext створюється та передає стратегію збереження файлу:
    - Спочатку застосовується TxtSaveStrategy, щоб зберегти файл у форматі .txt.
    - Потім використовується JsonSaveStrategy, щоб зберегти файл у форматі JSON.
    - Далі застосовується XmlSaveStrategy, щоб зберегти файл у форматі XML.
3. Виконання стратегії – викликається метод executeSave, який передає файл у відповідну стратегію:
    - TxtSaveStrategy – зберігає вміст файлу у простому текстовому форматі.
    - JsonSaveStrategy – конвертує вміст файлу у формат JSON за допомогою бібліотеки Gson і зберігає його.
    - XmlSaveStrategy – форматує вміст у формат XML і зберігає файл.

Приклад роботи патерну

1. Програма створює об'єкт файлу example з вмістом, кодуванням та назвою файлу.
2. Спочатку застосовується стратегія TxtSaveStrategy:
    - Файл зберігається як example.txt у вказаній директорії.
    - Вивід: "File saved as TXT: example".
3. Потім застосовується стратегія JsonSaveStrategy:
    - Файл конвертується у JSON та зберігається як example.json.
    - Вивід: "File saved as JSON: example".
4. Далі застосовується стратегія XmlSaveStrategy:

- Файл форматується як XML і зберігається як example.xml.
- Вивід: "File saved as XML: example".

Переваги використання шаблону Strategy:

1. Гнучкість – легко додати нові стратегії для збереження файлів у інших форматах (наприклад, YAML, CSV) без зміни коду контексту.
2. Розширюваність – додавання нової стратегії збереження вимагає створення лише нового класу, що реалізує інтерфейс SaveStrategy.
3. Чистий код – завдяки поліморфізму код стає зрозумілим і підтримуваним, оскільки вся логіка збереження файлів ізольована у відповідних стратегіях.

Висновок: У ході виконання лабораторної роботи я ознайомилася з теоретичними основами та практично реалізувала шаблон проєктування "Стратегія" для текстового редактора. Було створено діаграму класів, яка наочно демонструє роботу шаблону та взаємодію основних компонентів системи. Зокрема, інтерфейс SaveStrategy визначає єдиний метод save, а його конкретні реалізації – TxtSaveStrategy, JsonSaveStrategy та XmlSaveStrategy – забезпечують збереження файлів у різних форматах (TXT, JSON та XML).