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 №6

ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Виконав
студент групи ІА–22:
Щаблевський Д. Е.
Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

Тема: шаблони «Abstract Factory», «Factory Method», «Memento», «**Observer**», «Decorator»

Мета: ознайомитися з шаблонами проектування «Abstract Factory», «Factory Method», «Memento», «**Observer**», «Decorator», та набути практичних навичок їх застосування. Реалізувати частину функціоналу програми за допомогою одного з розглянутих шаблонів для досягнення конкретних функціональних можливостей та забезпечення ефективної взаємодії між класами.

Хід роботи

..28 JSON Tool (ENG) (strategy, command, observer, template method, flyweight)

Display JSON schema with syntax highlight. Validate JSON schema and display errors. Create user friendly table\list box\other for read and update JSON schema properties metadata (description, example, data type, format, etc.). Auto save\restore when edit, maybe history. Can check JSON value by schema (Put schema and JSON = valid\invalid, display errors). Export schema as markdown table. JSON to "flat" view.

Короткі теоретичні відомості

Шаблони проєктування представляють собою стандартизовані підходи до вирішення типових задач, які часто зустрічаються при створенні інформаційних систем. Вони включають в себе опис проблеми, ефективне рішення та рекомендації щодо застосування в різних контекстах. Завдяки загальновизнаним назвам, шаблони легко розпізнаються та багаторазово використовуються.

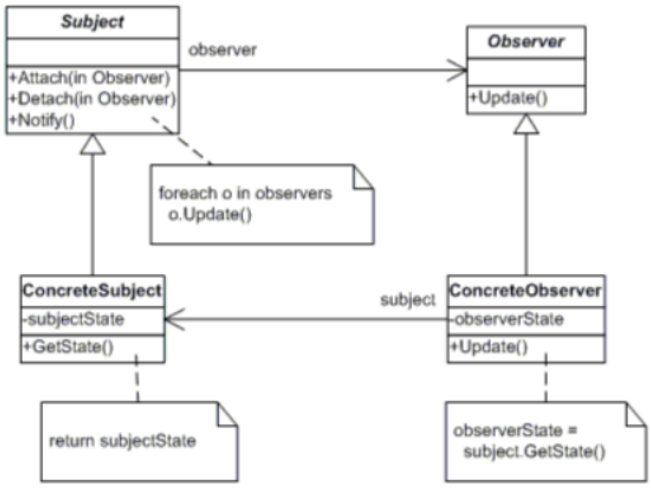
Ключовою особливістю роботи з шаблонами є ретельне моделювання предметної області, що дозволяє точно сформулювати проблему і підібрати найкращий шаблон. Використання таких рішень дає розробнику ряд переваг: вони допомагають зробити систему більш організованою, зрозумілою для вивчення та легкою у розширенні. Крім того, це підвищує гнучкість системи до змін і сприяє її простішій інтеграції з іншими рішеннями.

Таким чином, шаблони проєктування є перевіреними часом інструментами, які широко застосовуються в різних організаціях і проєктах для побудови ефективної архітектури в заданих умовах.

Навіщо використовувати шаблони ?

Шаблони проєктування сприяють зменшенню часу і зусиль на створення архітектури, надаючи системі важливі властивості, як-от гнучкість і адаптованість, а також спрощують її підтримку. Їх застосування полегшує комунікацію в команді, оскільки спільне розуміння шаблонів робить архітектуру зрозумілою для всіх учасників проєкту. Шаблони дозволяють уникнути "винаходу велосипеда", використовуючи перевірені практики, визнані спільнотою розробників, що економічно вигідно для бізнесу. Водночас їх використання повинно бути обґрунтованим, адже вони не є універсальним рішенням для всіх ситуацій.

Шаблон “Observer”



Призначення патерну Observer

Шаблон «Observer» (Спостерігач) дозволяє забезпечити автоматичне сповіщення залежних об'єктів про зміну стану об'єкта-наглядача. Наприклад, механізм підписки, де кілька об'єктів-підписників реагують на зміни в одному джерелі даних. Цей шаблон зручний для реалізації систем зі слабким зв'язуванням, де об'єкти мають залишатися незалежними, але синхронізованими.

Проблема

Спостерігач потрібен, якщо:

- Необхідно забезпечити автоматичне сповіщення та синхронізацію залежних об'єктів при зміні стану одного об'єкта.
- Потрібно підтримувати слабке зв'язування між об'єктами, щоб забезпечити масштабованість та гнучкість системи.

Рішення

Шаблон **«Спостерігач»** – це поведінковий шаблон проектування, який забезпечує механізм підписки, дозволяючи одним об'єктам відслідковувати та реагувати на зміни або події, що відбуваються в інших об'єктах. Реалізувати не менше 3-х класів відповідно до обраної теми

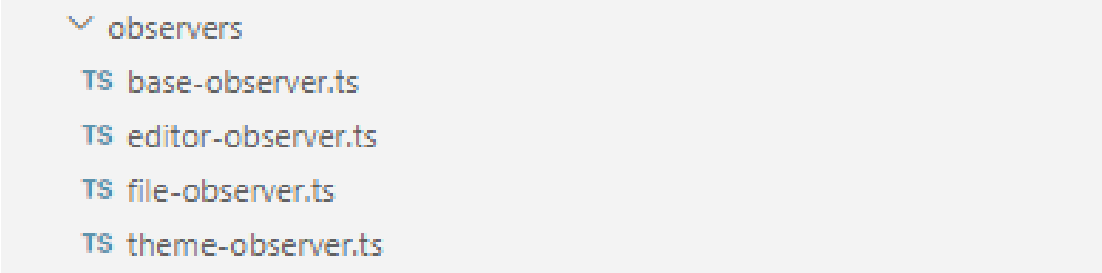


Рисунок 1 – структура класів

Опис класів

В ході виконання лабораторної роботи були реалізовані наступні класи:

- **Observer**
  - Інтерфейс, що визначає метод визначає контракт для спостерігачів, які отримують сповіщення від об'єкта Subject.
  - Має обов'язковий метод next(value: T) для отримання даних і необов'язковий метод error(error: Error) для обробки помилок.
- **Subject**
  - Клас для реалізації шаблону проєктування.
  - Зберігає список спостерігачів і повідомляє їх про зміни.

- **EditorStateSubject**
    - Клас-нащадок Subject.
    - Відповідає за відстеження змін стану редактора і сповіщення про них спостерігачів.
- **FileChangeSubject**
    - Клас-нащадок Subject.
    - Повідомляє спостерігачів про зміни файлів.

Реалізувати один з розглянутих шаблонів за обраною темою.

```
export interface Observer<T> {
  next(value: T): void;
  error?(error: Error): void;
}

export class Subject<T> {
  private observers: Observer<T>[] = [];

  subscribe(observer: Observer<T>): void {
    this.observers.push(observer);
  }

  next(value: T): void {
    this.observers.forEach(observer => observer.next(value));
  }

  error(error: Error): void {
    this.observers.forEach(observer => observer.error?.(error));
  }
}
```

```
export class EditorStateSubject extends Subject<string> {
  override next(state: string): void {
    new LoggerCommand('Editor Observer', 'Notifying state change to observers').execute();
    super.next(state);
  }
}
```

```
export class FileChangeSubject extends Subject<string> {
  override next(state: string): void {
    new LoggerCommand('File Observer', 'Notifying state change to observers').execute();
    super.next(state);
  }
}
```

Рисунок 2 – реалізований шаблон «Спостерігач»

Проблема, яку допоміг вирішити шаблон «Спостерігач»

Застосування патерну «Observer» вирішує проблему складного управління залежностями між об'єктами. Раніше, при зміні стану об'єкта, потрібно було вручну оновлювати всі залежні об'єкти, що ускладнювало підтримку коду і могло призвести до помилок. Завдяки патерну, залежні об'єкти (спостерігачі) автоматично отримують сповіщення про зміни, що спрощує синхронізацію та забезпечує гнучкість системи.

Переваги застосування патерну «Спостерігач»

1. **Гнучкість і масштабованість** - Нові спостерігачі можуть динамічно підписуватись на об'єкти Subject, забезпечуючи гнучке управління залежностями.
2. **Розділення відповідальності** - Спостерігачі (Observer) і об'єкти Subject мають чітко визначені ролі. Спостерігачі відповідають за реакцію на події, а Subject за сповіщення.
3. **Можливість розширення** - Завдяки наслідуванню (наприклад, EditorStateSubject, FileChangeSubject), можна легко створювати спеціалізовані об'єкти Subject для реагування на інші зміни у застосунку.

Діаграма класів для паттерну «Спостерігач»

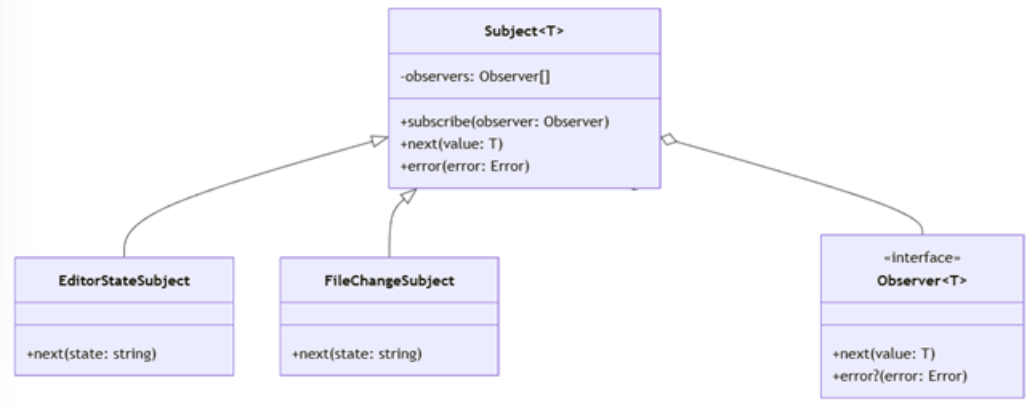


Рисунок 3 – діаграма класів

Посилання на репозиторій з кодом: https://github.com/neodavis/json-tool

Висновок: У ході виконання лабораторної роботи було реалізовано патерн проєктування Observer для функціоналу сповіщення підписників про зміни у стані об'єкта. Шаблон «Спостерігач» забезпечив автоматичну синхронізацію залежних об'єктів під час виконання програми. Це зробило систему більш гнучкою, масштабованою та простою для розширення й підтримки.