Чернівецький національний університет імені Юрія Федьковича

Факультет математики та інформатики

Кафедра математичного моделювання

**Лабораторна робота №4**

з навчальної дисципліни: “Проектування програмних систем”

тема: “**Застосування патернів програмування.**”

Виконав: студент 3-го курсу

301-Б групи

спеціальності “Комп'ютерні науки”

Пандаров В.О.

**Чернівці – 2024**

**Варіант №16**

**Завдання.**

1. Аналізуючи діаграми класів (Лабораторна робота №3), обґрунтувати

застосування підібраних для реалізації патернів.

2. Використовуючи лабораторні роботи із дисциплін професійної

підготовки, створити репозиторій, розмістити файли проектів. Надати

доступ викладачам на github / gitlab ресурсі.

**Завдання 1**

Застосування патернів проектування в розробці програмного забезпечення дозволяє розв’язати типові проблеми та зробити систему більш гнучкою, масштабованою та легко підтримуваною.

Розглянемо загальні обґрунтування для вибору патернів проектування, що були рекомендовані для діаграм:

**1. Патерн Станів (State Pattern)**

Застосування:

- Використовується для того, щоб дозволити об'єкту змінювати свою поведінку при зміні його внутрішнього стану.

- Кожен стан реалізується як окремий клас, що дозволяє легко додавати нові стани або змінювати існуючі без великих змін у коді.

Переваги:

- Покращує читабельність та організацію коду.

- Спрощує додавання нових станів.

- Локалізує код, пов'язаний з конкретними станами, що полегшує його тестування і підтримку.

**2. Патерн Ланцюгу обов'язків (Chain of Responsibility Pattern)**

Застосування:

- Дозволяє передавати запити вздовж ланцюга обробників, де кожен обробник вирішує, чи обробляти запит самостійно, чи передати його наступному обробнику.

- Корисний для реалізації послідовності дій, які можуть оброблятися різними компонентами.

Переваги:

- Зменшує зв’язність між об’єктами-обробниками.

- Дозволяє легко додавати нові обробники або змінювати порядок обробки.

- Покращує гнучкість і розширюваність системи.

**3. Патерн Шаблонних методів (Template Method Pattern)**

Застосування:

- Визначає скелет алгоритму в методі, залишаючи реалізацію деяких кроків підкласам.

- Забезпечує контроль над структурою алгоритму, дозволяючи змінювати деталі реалізації без змін основної логіки.

Переваги:

- Сприяє повторному використанню коду.

- Спрощує підтримку і розширення алгоритмів.

- Зменшує дублювання коду, оскільки спільні частини алгоритму визначаються в базовому класі.

**4. Патерн Стратегії (Strategy Pattern)**

Застосування:

- Дозволяє визначити сімейство алгоритмів, інкапсулювати кожен з них та зробити їх взаємозамінними.

- Дозволяє варіювати алгоритми незалежно від клієнтів, які ними користуються.

Переваги:

- Спрощує заміну та розширення алгоритмів.

- Зменшує дублювання коду.

- Підвищує гнучкість програми за рахунок вибору алгоритму під час виконання.

**5. Патерн Фасаду (Facade Pattern)**

Застосування:

- Надає уніфікований інтерфейс до набору інтерфейсів у підсистемі, полегшуючи використання підсистеми.

- Приховує складність системи, надаючи простіший інтерфейс.

Переваги:

- Зменшує кількість залежностей між підсистемами та клієнтами.

- Полегшує використання складних систем, надаючи простий інтерфейс.

- Покращує модульність і розширюваність системи.

**6. Патерн Команди (Command Pattern)**

Застосування:

- Інкапсулює запит як об’єкт, дозволяючи параметризувати клієнтів з різними запитами, чергувати або реєструвати запити та підтримувати скасування операцій.

Переваги:

- Збільшує гнучкість при виконанні запитів.

- Дозволяє легко реалізовувати черги запитів та механізми скасування.

- Сприяє зниженню зв'язності між об'єктами, що викликають операції, та об'єктами, що їх виконують.

**7. Патерн Спостерігача (Observer Pattern)**

Застосування:

- Визначає залежність "один до багатьох" між об'єктами, так що при зміні стану одного об'єкта всі залежні об'єкти сповіщаються та автоматично оновлюються.

Переваги:

- Забезпечує автоматичну синхронізацію об'єктів.

- Зменшує зв'язність між об'єктами, що дозволяє їм взаємодіяти незалежно.

- Підвищує розширюваність системи за рахунок додавання нових спостерігачів без зміни об'єкта-наблюдача.

**Висновок:**

Застосування цих патернів проектування до системи управління готелем, процесу онлайн-бронювання та процесу реєстрації гостей покращує структуру коду, робить його більш гнучким та полегшує підтримку. Патерни дозволяють зменшити зв’язність між компонентами, забезпечити повторне використання коду та полегшити додавання нових функцій.

**Завдання 2**

<https://github.com/Vlad-Panda/PPS.git>

