Чернівецький національний університет імені Юрія Федьковича
Факультет математики та інформатики
Кафедра математичного моделювання

Лабораторна робота №4

«Застосування патернів програмування»

Виконав: Кирлан Іван Джованні
Системний аналіз
307 група
Викладач: Піддубна Л.А

Чернівці 2024

1. Аналізуючи діаграми класів (Лабораторна робота №3), обґрунтувати застосування підібраних для реалізації патернів

- **Facade**

Патерн **Facade** забезпечує єдину спрощену точку доступу до підсистеми. Клас **Interface** служить цією спрощеною точкою доступу до системи, де він має методи для отримання вводу, відображення повідомлень та управління транзакціями (**getInput()**, **displayMessage()**, **startTransaction()**, **endTransaction()**).

Використання патерну Facade допомагає зменшити складність взаємодії з системою для користувачів або інших систем, приховуючи деталі реалізації і надаючи чітко визначені методи для основних операцій.

- **Composite**

Патерн **Composite** дозволяє групувати об'єкти в деревоподібні структури для представлення ієрархії частин та цілих. Клас **Ticket** має методи для отримання інформації про квиток, резервування місць та валідації квитків (**getTicketInformation()**,**reserveSeat()**, **validateTicket()**).

Клас **Ticket** може розглядатися як компонент, який є частиною більших структур квиткової системи. Використання **Composite** дозволяє об'єднати квитки в більш складні структури і обробляти їх як окремі об'єкти або колекції об'єктів.

- **Strategy**

Патерн **Strategy** дозволяє визначати сімейство алгоритмів, інкапсулювати їх та робити їх взаємозамінними. Клас **Payment** має методи для обробки та валідації платежів (**processPayment()**, **validatePayment()**,

refundPayment()), що можуть реалізовувати різні стратегії оплати.

Використання патерну **Strategy** дозволяє класу **Payment** підтримувати різні способи оплати та механізми валідації, роблячи їх взаємозамінними та розширюваними без зміни основної логіки класу.

- **Observer**

Патерн **Observer** дозволяє об'єктам сповіщати інші об'єкти про зміни свого стану. Клас **Manager** має методи для перегляду операцій, вирішення помилок та обробки повернень (**viewOperations()**, **resolveErrors()**, **handleReturns()**), що можуть бути сповіщені про зміни в системі.

Використання патерну **Observer** дозволяє класу **Manager** реагувати на зміни в системі, такі як нові транзакції або помилки, та виконувати відповідні дії, зберігаючи при цьому слабке зчеплення між класами.

- **Factory Method**

Патерн **Factory Method** надає інтерфейс для створення об'єктів, але дозволяє підкласам змінювати тип створюваного об'єкта. Клас **Theater Show** містить методи для отримання деталей шоу та переліку шоу (**getShowDetails()**, **listShows()**), що можуть використовувати фабричний метод для створення різних видів шоу.

Використання патерну **Factory Method** дозволяє гнучко створювати різні види театральних шоу, що може бути корисним при розширенні системи та додаванні нових типів шоу в майбутньому.

2. Використовуючи лабораторні роботи із дисциплін професійної підготовки, створити репозиторій, розмістити файли проектів. Надати доступ викладачам на github / gitlab ресурсі.

https://github.com/IvanKyrlan/design-of-software-systems.git