**Лабораторна робота №4**

**«Розробка технічного завдання програмної системи»**

**Підготував Крюков Владислав 311 гр.**

**Тема: Залізничний вокзал**

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

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

Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис

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

**Спостерігач (Observer):**

**Проблема, що виникає:**

В системі залізничного вокзалу пасажири часто бажають отримувати сповіщення про зміни, які відбуваються у системі, такі як зміни в розкладі поїздів, доступність квитків на певні маршрути тощо. Без механізму сповіщень пасажирам доводиться постійно перевіряти систему на наявність оновлень, що може бути незручно та витратно з точки зору часу.

**Рішення, яке пропонується:**

Використання патерна "Спостерігач" дозволяє пасажирам підписатися на отримання сповіщень про зміни у системі та автоматично отримувати оновлення без необхідності постійної перевірки системи. У цьому патерні пасажири, які бажають отримувати сповіщення, реєструються як спостерігачі, а система залізничного вокзалу виступає як об'єкт спостереження. Коли в системі відбувається зміна, вона автоматично повідомляє всіх своїх спостерігачів про цю зміну, і кожен спостерігач може реагувати відповідно до своїх потреб.

Наприклад, пасажир може підписатися на отримання сповіщень про зміни у розкладі свого поїзда. Коли буде змінено розклад, система автоматично надсилатиме сповіщення цьому пасажирові з оновленою інформацією про час відправлення поїзда. Це дозволяє пасажиру бути завжди в курсі останніх змін і забезпечує кращу взаємодію з системою залізничного вокзалу.

**Переваги використання патерна "Спостерігач" у системі залізничного вокзалу:**

* **Розділення обов'язків**: За допомогою цього патерна система стає більш модульною, оскільки реалізація спостерігача відділена від логіки, що викликає оновлення. Це дозволяє зберігати чистоту коду та зменшувати залежності між компонентами.
* **Гнучкість та розширюваність:** Спостерігачі можуть бути додані або видалені з системи без значних змін в інших частинах системи. Це забезпечує гнучкість у внесенні змін та розширенні функціональності.
* **Уникнення циклічних залежностей:** Використання патерна "Спостерігач" допомагає уникнути ситуацій, коли компоненти системи залежать один від одного напряму, що може призвести до циклічних залежностей та складних проблем з обслуговуванням коду.
* **Підтримка реактивного програмування:** Цей патерн добре поєднується з парадигмою реактивного програмування, де об'єкти реагують на зміни стану і автоматично сповіщають своїх спостерігачів про ці зміни.
* **Зручний інтерфейс для споживачів інформації:** Пасажири можуть легко підписатися на отримання сповіщень про зміни у системі, що забезпечує їм зручний та швидкий спосіб отримання актуальної інформації без постійного моніторингу системи.
* **Зменшення навантаження на сервер:** Система може ефективно відправляти сповіщення лише тим спостерігачам, які підписані на конкретні події, що дозволяє зменшити навантаження на сервер та забезпечує ефективне використання ресурсів.

**Ланцюжок обов'язків (Chain of Responsibility):**

**Проблема:**

У системі залізничного вокзалу часто виникають ситуації, коли запити користувачів можуть оброблятися кількома різними об'єктами, кожен з яких має свої умови для обробки. Наприклад, при бронюванні квитка можуть знадобитися різні перевірки, такі як наявність місць, вірність введених даних тощо.

**Рішення:**

Патерн «Ланцюжок обов'язків» дозволяє організувати послідовність об'єктів-обробників, кожен з яких здатний обробляти певний тип запиту. Запит проходить через ланцюжок, поки не знайдеться об'єкт, який може його обробити. Кожен об'єкт у цьому ланцюжку має можливість або обробити запит, або передати його наступному об'єкту.

**Переваги:**

* **Гнучкість і розширюваність:** Патерн дозволяє легко додавати нові об'єкти-обробники без змін в існуючому коді, роблячи систему більш гнучкою.
* **Чітке розділення обов'язків:** Кожен об'єкт у ланцюжку відповідає за обробку певного типу запиту, що знижує залежності між класами та сприяє кращій організації коду.
* **Принцип відкритості/закритості:** Нові об'єкти можуть бути додані в ланцюжок без змін існуючого коду, що забезпечує стабільність та гнучкість системи.
* **Уникнення жорстких зв’язків:** Об'єкти у ланцюжку залежать тільки від наступного об'єкта, що дозволяє уникнути тісної інтеграції між компонентами.

**Прототип (Prototype):**

**Проблема:**

У системі залізничного вокзалу часто потрібно створювати нові об'єкти, схожі за структурою або властивостями на існуючі, але з деякими змінами. Наприклад, при повторному бронюванні квитка, де змінюється лише дата. Створення нових об'єктів з нуля може бути трудомістким та призводити до повторення коду.

**Рішення:**

Патерн "Прототип" дозволяє створювати нові об'єкти шляхом копіювання існуючих (прототипів) і зміни необхідних атрибутів. Замість створення кожного нового об'єкта з нуля, можна просто скопіювати прототип і внести потрібні зміни.

**Переваги:**

* **Економія часу та ресурсів:** Використання патерна "Прототип" допомагає уникнути повторного написання коду для створення кожного нового об'єкта, що зберігає час і зменшує кількість помилок.
* **Зручність і простота:** Копіювання та модифікація існуючих об'єктів дозволяє швидко створювати нові об'єкти з потрібними атрибутами, що полегшує роботу розробників.
* **Збереження цілісності структури:** Оскільки нові об'єкти створюються на основі існуючих прототипів, це гарантує збереження структурної цілісності та уникнення помилок при створенні об'єктів.
* **Підтримка розширення:** Простота і гнучкість патерна "Прототип" дозволяють легко змінювати та розширювати систему без значних зусиль, сприяючи підтримці та розвитку проекту.

**Патерн Білдер (Builder):**

**Проблема:**

У системі залізничного вокзалу існує потреба у створенні об'єктів зі складною структурою, таких як квиток на поїзд з різними додатковими опціями (наприклад, харчування, місце з підзарядкою, постільна білизна тощо). Використання звичайного конструктора для створення таких об'єктів може бути незручним або навіть неможливим через велику кількість параметрів.

**Рішення:**

Патерн Білдер дозволяє розділити процес створення складного об'єкта на окремі кроки, надаючи можливість побудувати об'єкт поетапно. Клас квитка може мати Білдер, який має методи для додавання різних додаткових опцій, таких як харчування, місце з підзарядкою тощо. Користувач може викликати ці методи для додавання або вилучення певних опцій до квитка.

**Переваги:**

* **Гнучкість у створенні об'єктів:** Патерн «Білдер» дозволяє створювати різні варіації об'єктів залежно від потреб користувача шляхом додавання або вилучення певних характеристик.
* **Зрозумілий інтерфейс:** Використання «Білдера» робить процес створення об'єктів більш зрозумілим і структурованим, оскільки кожен метод відповідає за певну характеристику об'єкта.
* **Уникнення перевантаження конструктора:** Патерн «Білдер» дозволяє уникнути перевантаження конструктора об'єкта, особливо коли є багато параметрів або можливих комбінацій параметрів.
* **Підтримка інкапсуляції:** «Білдер» дозволяє інкапсулювати процес створення складного об'єкта в одному місці, що спрощує внесення змін та розширення логіки створення.