# **Лабораторна робота №2: Вивчення патернів проектування**

## **Мета роботи**

Метою виконання лабораторної роботи №2 є вивчення патернів проектування, зокрема класичних патернів GoF (Gang of Four) з книги "Design Patterns: Elements of Reusable Object-Oriented Software" авторів Erich Gamma, Richard Helm, Ralph Johnson, і John Vlissides. При цьому корисно вивчати не лише «штучні» приклади патернів, але приклади їх використання в контексті якогось відносно великого проєкту.

## **Завдання**

1. Реалізувати консольну програму, яка використовує патерни Strategy, Singleton, Builder, Decorator, та Proxy.
2. Забезпечити, щоб кожен патерн був інтегрований у функціонал програми.
3. Продемонструвати роботу програми за допомогою взаємодії користувача через консоль.

## **Опис програми**

### **Основні класи та їх взаємодія**

#### **TaskManager (Патерн Singleton)**

**Опис:** Цей клас відповідає за управління завданнями в системі. Він гарантує, що існує лише один екземпляр TaskManager, який використовується для додавання, збереження та відправки завдань.

**Взаємодія:** TaskManager управляе всіма завданнями та забезпечує їх відправку через проксі.

#### **Task (Патерн Builder)**

**Опис:** Клас Task представляє навчальне завдання, яке має питання і відповідь. Використовуючи внутрішній клас Builder, можна гнучко створювати завдання з різними параметрами.

**Взаємодія:** Завдання створюються через Builder, що забезпечує зручний і зрозумілий спосіб їх побудови.

#### **TaskDecorator (Патерн Decorator)**

**Опис:** Абстрактний клас, який дозволяє додавати додаткові умови або характеристики до завдань. Це дозволяє розширювати функціональність завдань без зміни їх основного коду.

**Взаємодія:** Декоратори обгортають завдання і додають до них нові можливості, такі як обмеження по часу.

#### **TimeLimitDecorator (Патерн Decorator)**

**Опис:** Конкретний декоратор, який додає до завдання обмеження по часу. Він модифікує питаннч завдання, додаючи інформацію про час.

**Взаємодія:** Використовується для додавання тимчасових обмежень до завдань, створених через Task.

#### **ScoreStrategy (Патерн Strategy)**

**Опис:** Інтерфейс, який визначає метод для підрахунку балів на основі відповіді студента. Різні реалізації цього інтерфейсу можуть використовувати різні алгоритми підрахунку балів.

**Взаємодія:** Стратегія підрахунку балів використовується для визначення оцінки відповіді студента на завдання.

#### **ExactMatchScoreStrategy (Патерн Strategy)**

**Опис:** Конкретна реалізація інтерфейсу ScoreStrategy, яка перевіряє, чи відповідає відповідь студента точній відповіді завдання. Якщо відповіді співпадають, студент отримує один бал, інакше - нуль балів.

**Взаємодія:** Використовується для підрахунку балів на основі точного співпадіння відповідей.

#### **TaskSender (Патерн Proxy)**

**Опис:** Інтерфейс, який визначає метод для відправки завдань на віддалений сервер або інший зовнішній сервіс.

**Взаємодія:** Використовується для відправки завдань через проксі.

#### **TaskSenderProxy (Патерн Proxy)**

**Опис:** Клас-проксі, який реалізує інтерфейс TaskSender і виконує відправку завдань через RemoteTaskSender. Він додає рівень абстракції для взаємодїї із зовнішніми сервісами.

**Взаємодія:** Використовується для відправки завдань на віддалений сервер, забезпечуючи додатковий рівень контролю та безпеки.

#### **RemoteTaskSender (Патерн Proxy)**

**Опис:** Клас, який безпосередньо відправляє завдання на віддалений сервер, наприклад, за допомогою HTTP-запиту. Він реалізує інтерфейс TaskSender.

**Взаємодія:** Використовується проксі для реальної відправки даних до віддаленого сервера.

## **Висновок**

Ця програма демонструє, як використовувати патерни проектування для створення гнучкої, масштабованої та розширюваної системи управління навчальними завданнями.

* **Strategy:** Інтерфейс ScoreStrategy та його реалізація ExactMatchScoreStrategy використовуються для реалізації різних стратегій підрахунку балів. Це дозволяє легко змінювати алгоритм підрахунку балів.
* **Singleton:** Клас TaskManager є Singleton, що гарантує існування лише одного екземпляра цього класу. Він управляє всіма завданнями в системі.
* **Builder:** Клас Task використовує внутрішній клас TaskBuilder для побудови складних завдань. Це дозволяє створювати завдання з різними параметрами в гнучкий і читабельний спосіб.
* **Decorator:** Абстрактний клас TaskDecorator та його конкретна реалізація TimeLimitDecorator дозволяють додавати додаткові умови до завдань. Це забезпечує динамічне розширення функціональності завдань без зміни їх основного коду.
* **Proxy:** Клас TaskSenderProxy реалізує інтерфейс TaskSender і використовує RemoteTaskSender для відправки завдань до віддаленого сервера. Це дозволяє додати рівень абстракції для взаємодії із зовнішніми сервісами.

Програма є прикладом використання патернів проектування у відносно великому проекті, що робить її код більш структурованим, гнучким і легким для підтримки та модифікації.

P.s. Оскільки проект не завершений і перебуває в процесі доопрацювання, будуть вноситися відповідні зміни до коду та звітів.