ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №5

за курсом «Об’єктно Орієнтованого Програмування»

студента групи ПЗ-21-2

Гудимова Дениса Сергійовича

кафедра математичного забезпечення ЕОМ, ДНУ

2022/2023 навч.р.

Постановка задачі

Реалізувати Паттерн Decorator та Паттерн Repository (Репозиторій)

Опис розв’язку

Для реалізації патерну Decorator з курсової роботи було залишено фігури квадрат та коло. На основі абстрактного класу(інтерфейсу) Figure був створений клас FigureDecorator який реалізує усі методи абстрактного класу і має поле типу Figure. Також було створено 3 класи : InternalColorDecorator, OutlineColorDecorator, RotateDecorator які успадковують клас FigureDecorator у кожному класі конструктор реалізований таким чином що параметр це абстрактний клас Figure, а також перевизначено метод change\_color(sf::Color color). Цей метод у всіх 3 класах Decorator реалізований таким чином, що він робить свою задачу незалежно від інших і викликає цей же метод для вкладеного об’єкту і так доки цей метод послідовно не відпрацює для усіх об’єктів які загорнуті у середину.

Для реалізації патерну Repository було створено новий проект у якому створені класи User який зберігає інформацію про користувача, FileWriter котрий записує у json файл інформацію про користувача, Repository абстрактний клас, UserRepository наслідує Repository та реалізує усі методи для додавання та пошуку користувача. Клас UserRepository створюєтьяс для того щоб не порушувати концепт ООП і не повторювати поширені запроси і не порушувати концепт DRY коли потрібно викликати запроси із різних місць.

Вихідний текст програми розв’язку задачі

Текст розв’язку для задачі (рис. 3.1 - 3.13).

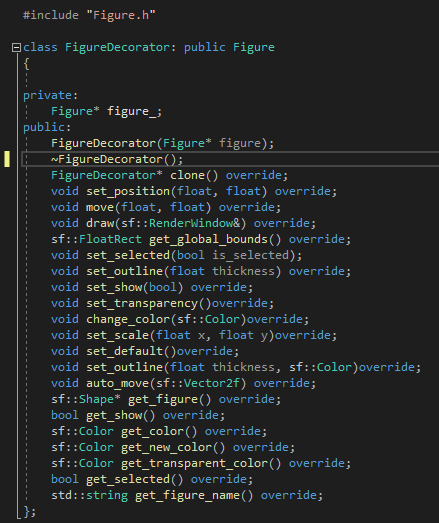


Рисунок 3.1 – Реалізація патерну Decorator

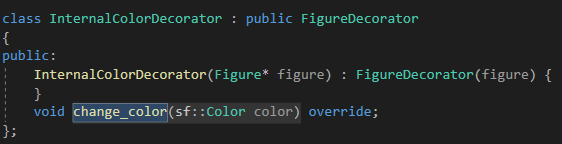


Рисунок 3.2 – Ініціалізація InternalColorDecorator

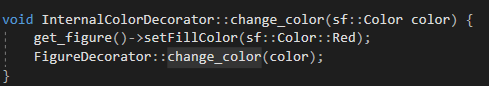


Рисунок 3.3 – Реалізація InternalColorDecorator

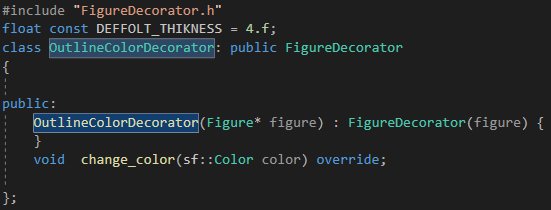


Рисунок 3.4 – Ініціалізація OutlineColorDecorator

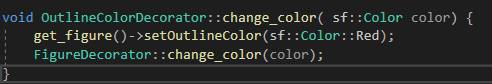


Рисунок 3.5 – Реалізація OutlineColorDecorator

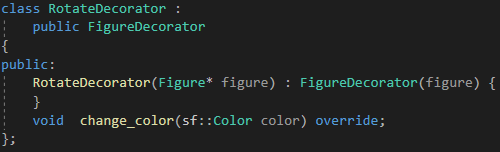


Рисунок 3.6 – Ініціалізація RotateDecorator

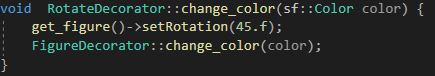


Рисунок 3.7 – Реалізація RotateDecorator

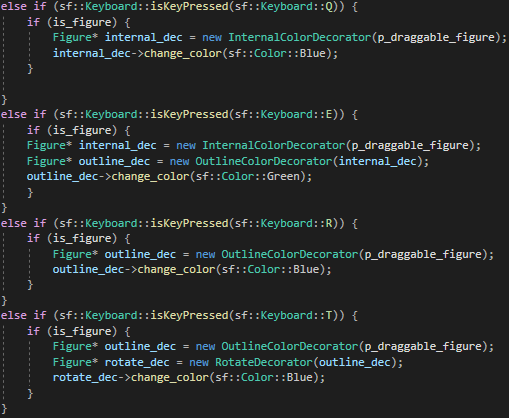


Рисунок 3.8 – Використання Decorators для аибраної фігури

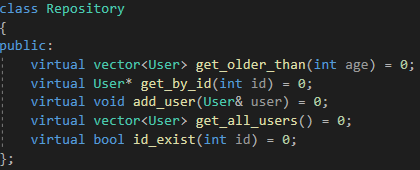


Рисунок 3.9 – Реалізація інтерфеса Repository

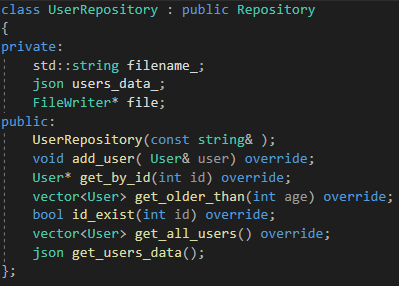


Рисунок 3.10 – Наслідуваня інтерфеса Repository

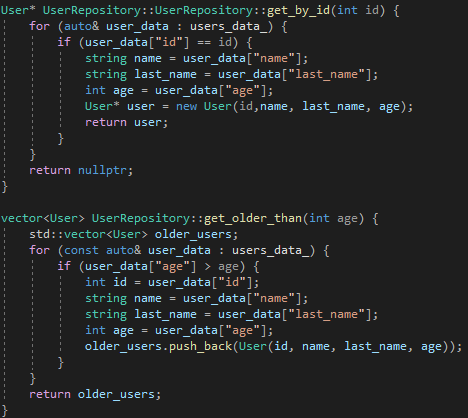


Рисунок 3.11 – Реалізація методів «get\_by\_id» та «get\_older\_than» для User

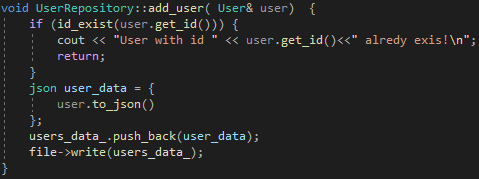


Рисунок 3.12 – Реалізація методу «add\_user» для User

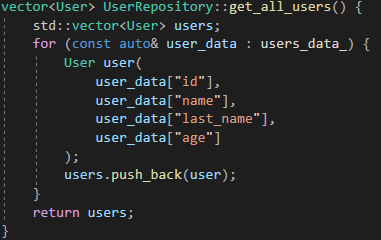


Рисунок 3.13 – Реалізація методу «get\_all\_users» для User

Опис іинтерфейсу (керівництво користувача)

Для патерну Decorator користувач натискає:

1. Q щоб загорнути вибраний об’єкт у внутрішній декоратор і змінити колір;
2. E щоб спочатку загорнути в внутрішній декоратор а потім у зовнішній і змінити колір ;
3. R щоб спочатку загорнути у зовнішній декоратор і змінити колір ;
4. T щоб спочатку загорнути у зовнішній декоратор а потім у повертаючий і змінити колір.

Для патерну Repository користувач у консольному застосунку вибирає дію яку він хоче зробити (рис. 4.1):

Додавання нового користувача, потрібно ввести id, ім’я, прізвище та вік;

1. Знайти користувача за id;
2. Вивести у консоль інформацію про користувачів які старші за введений вік;
3. Вивести інформацію про усіх користувачів;
4. Завершити програму.

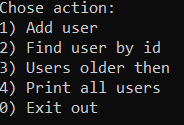


Рисунок 4.1 – Користувацький інтерфейс для Repository

Опис тестових прикладів

Тестові приклади для Decorator (рис. 5.1- 5.8).

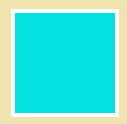


Рисунок 5.1 – Початковий стан



Рисунок 5.2 – Після натискання Q

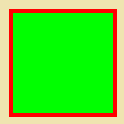


Рисунок 5.3 – Після натискання E

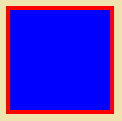


Рисунок 5.4 – Після натискання R

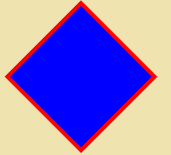


Рисунок 5.5 – Після натискання T

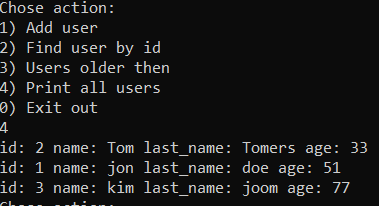


Рисунок 5.6 – Вивід у консоль вміст Repository



Рисунок 5.7 – Вміст Repository у json файлі

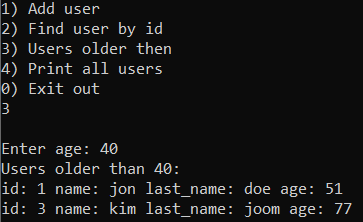


Рисунок 5.8 – Вивід у консоль користувачів старших за 40

1. Аналіз помилок (опис усунення зауважень)

Звіт відформатовано