Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Технології розроблення програмного забезпечення

Лабораторна робота №4

ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»

Виконала
студентка групи ІА–24:
Кармазіна А. В.

Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Завдання

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для
досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

Тема: 26 Download manager (iterator, command, observer, template method, composite, p2p)

Інструмент для скачування файлів з інтернету по протоколах http або https з можливістю продовження завантаження в зупиненому місці, розподілу швидкостей активним завантаженням, ведення статистики завантажень, інтеграції в основні браузери (firefox, opera, internet explorer, chrome) (https://github.com/Karmazinanastya/TRPZ_Karmazina_labs/tree/main/FileDownloader)

Теоретичні відомості

Adapter забезпечує взаємодію між класами з несумісними інтерфейсами, перетворюючи інтерфейс одного класу в інтерфейс, зрозумілий іншому.

Builder розділяє процес створення складного об'єкта на окремі кроки, дозволяючи поступово збирати об'єкт, варіюючи його конфігурацію без зміни коду.

Command інкапсулює запити або операції в окремі об’єкти, дозволяючи параметризувати клієнтські класи і створювати черги операцій або скасовувати виконані дії.

Chain of Responsibility організовує послідовність обробників, через які проходить запит, поки його не буде оброблено відповідним обробником, що сприяє зменшенню жорсткої прив’язки між відправником і отримувачем.

Prototype дозволяє створювати нові об'єкти шляхом клонування існуючих, що забезпечує ефективність створення складних об'єктів, мінімізуючи витрати на їх ініціалізацію.

Хід роботи

Реалізація шаблону Builder

Шаблон проектування Builder використовується у класі HistoryBuilder, який допомагає будувати об'єкт типу History за допомогою послідовних викликів методів для налаштування його властивостей. Ось як цей шаблон працює:

Шаблон Builder дозволяє створювати складні об'єкти, крок за кроком, де кожен крок є окремим методом. Кожен метод модифікує один аспект об'єкта і повертає поточний об'єкт Builder для підтримки ланцюгового виклику методів. Це дозволяє розділити процес побудови об'єкта від його конкретної репрезентації, що робить код більш гнучким і легким для підтримки.

```
using FileDownloader.Data.Models;

namespace FileDownloader.UI.HistoryBuilders
{
    9 references
    public class HistoryBuilder
    {
        private History _history;

        1 reference
        public HistoryBuilder()
        {
            _history = new History();
        }

        1 reference
        public HistoryBuilder SetId(int id)
        {
            _history.Id = id;
            return this;
        }

        1 reference
        public HistoryBuilder SetFileName(string fileName)
        {
            _history.FileName = fileName;
            return this;
        }

        1 reference
        public HistoryBuilder SetFilePath(string filePath)
        {
            _history.FilePath = filePath;
            return this;
```

No issues found

Рис. 1 - Клас HistoryBuilder

1. Клас HistoryBuilder має приватну змінну _history, яка є об'єктом типу History.
2. Кожен метод, наприклад SetId(), SetFileName() і т.д., відповідає за встановлення одного значення у властивості об'єкта History.
3. Метод кожного виклику повертає поточний об'єкт Builder (this), що дозволяє викликати методи в ланцюгу.
4. Коли всі потрібні властивості об'єкта задані, викликається метод Build(), який повертає готовий об'єкт History.

Висновок

Шаблон Builder є потужним інструментом для створення складних об'єктів у програмуванні, дозволяючи відокремити процес побудови об'єкта від його репрезентації. У наведеному коді клас HistoryBuilder забезпечує крокове налаштування властивостей об'єкта History через ланцюгові виклики методів. Це дозволяє легко модифікувати тільки ті властивості, які необхідні, і забезпечує зручний спосіб створення об'єктів без необхідності використання великих конструкторів або великих блоків коду.