Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 8

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконала:
студентка групи ІА-31
Кизим Є. К.

Київ 2025

Зміст

Мета: Вивчити структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчитися застосовувати їх в реалізації програмної системи.

Завдання:

• Ознайомитись з короткими теоретичними відомостями.

• Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

• Реалізувати один з розглянутих шаблонів за обраною темою.

• Реалізувати не менше 3-х класів відповідно до обраної теми.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

Моя тема: 27. Особиста бухгалтерія (state, prototype, decorator, bridge, flyweight, SOA) Програма повинна бути наочним засобом для ведення особистих фінансів: витрат і прибутку; з можливістю встановлення періодичних витрат / прибутку (зарплата і орендна плата); введення сканованих чеків з відповідними статтями витрат; побудова статистики; експорт/імпорт в Excel, реляційні джерела даних; різні рахунки; ведення єдиного фонду на всі рахунки (всією сім'єю) – на особливі потреби (ремонт, автомобіль, відпустка); можливість введення вкладів / кредитів для контролю банківських рахунків (звірка нарахованих відсотків з необхідними і т.д.).

Теоретичні відомості

1. **Детальний опис досліджуваних шаблонів**

- **Composite (Компонувальник)**
  - Призначення: Об'єднує об'єкти в деревоподібні структури, дозволяючи клієнтам працювати з одиничними об'єктами (листами) та їхніми колекціями (контейнерами) однаково .

- Принцип роботи: Визначається спільний інтерфейс (Component) для всіх об'єктів. Контейнер (Composite) зберігає посилання на інші компоненти і делегує виконання спільних операцій (наприклад, розрахунок балансу) своїм дочірнім елементам, рекурсивно підсумовуючи результати.

- **Flyweight (Полегшувач)**
  - Призначення: Мінімізує використання пам'яті шляхом спільного використання максимальної кількості даних між багатьма об'єктами .
  - Застосування: Ефективний для роботи з великою кількістю об'єктів, які мають багато спільних (незмінних) даних, залишаючи лише унікальні (мінливі) дані індивідуальними.
- **Interpreter (Інтерпретатор)**
  - Призначення: Визначає, як інтерпретувати речення певної мови за допомогою ієрархії класів .
  - Застосування: Використовується для створення парсерів або обробників простих виразів (наприклад, регулярних виразів, SQL-запитів).
- **Visitor (Відвідувач)**
  - Призначення: Дозволяє відокремити новий алгоритм або операцію від об'єктів, над якими він працює, не змінюючи класи цих об'єктів .
  - Принцип роботи: Відвідувач містить метод Visit() для кожного типу об'єкта у структурі.

2. Обґрунтування вибору шаблону для реалізації

Для вдосконалення архітектури системи «Особиста бухгалтерія» було обрано структурний шаблон **Composite (Компонувальник)**. Це дозволило реалізувати вимогу щодо **ведення єдиного фонду** та розрахунку загального капіталу . Впровадження IFinanceComponent і класу UserAssets (як контейнера) дало змогу розрахувати загальний капітал користувача за допомогою єдиного виклику, незважаючи на те, що активи фізично є різними класами (Account та Fund).

Хід роботи

Діаграма класів реалізації патерну Composite

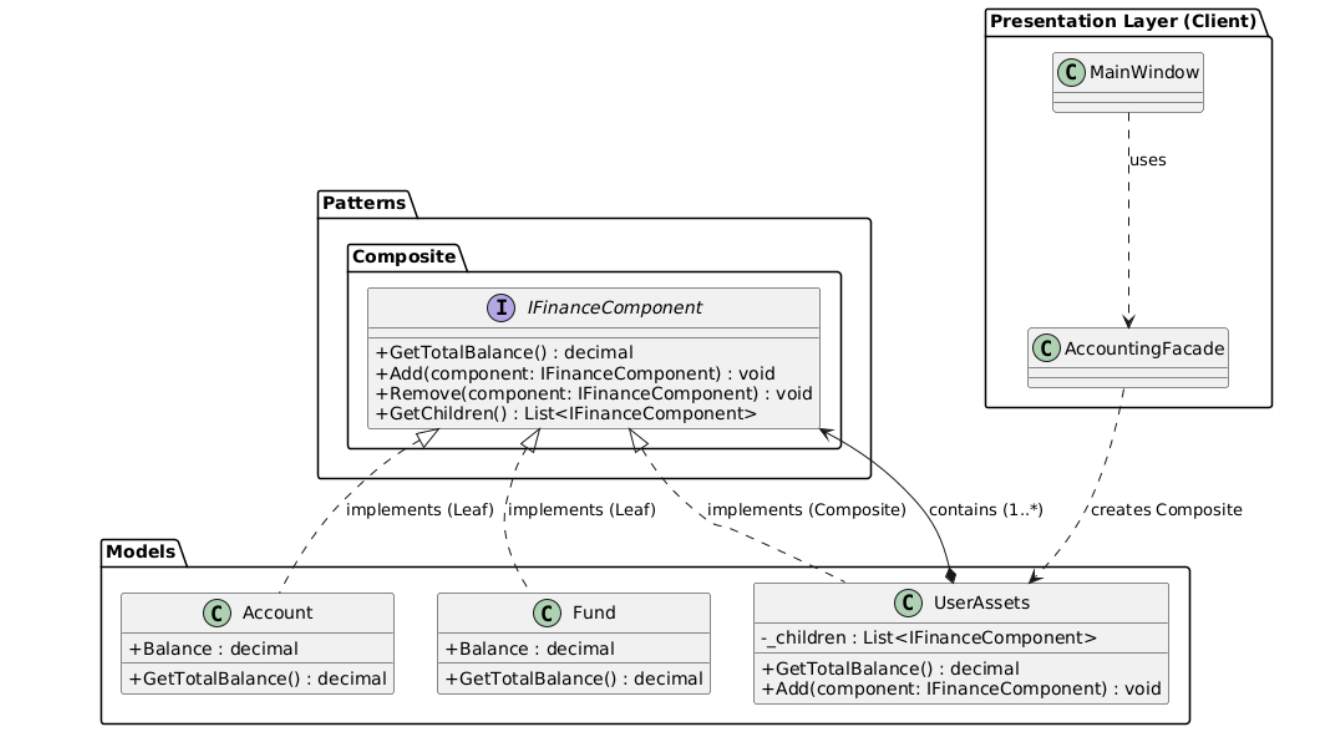


Рисунок 1 – Діаграма класів реалізації патерну Composite

Опис реалізації патерну «Composite»:

Фрагменти програмного коду:

Інтерфейс компонента (Patterns/Composite/IFinanceComponent.cs):

```
using personalAccounting.Models;
using System.Collections.Generic;

namespace personalAccounting.Patterns.Composite
{
    public interface IFinanceComponent
```

```
    {
        decimal GetTotalBalance();
        void Add(IFinanceComponent component);
        void Remove(IFinanceComponent component);
        List<IFinanceComponent> GetChildren();
    }
}
```

Контейнерний компонент (Models/UserAssets.cs):

```
using personalAccounting.Patterns.Composite;
using System.Collections.Generic;
using System.Linq;

namespace personalAccounting.Models
{
    public class UserAssets : IFinanceComponent
    {
        private List<IFinanceComponent> _children = new List<IFinanceComponent>();

        public string UserName { get; set; }

        public decimal GetTotalBalance()
        {
            return _children.Sum(c => c.GetTotalBalance());
        }

        public void Add(IFinanceComponent component)
        {
            _children.Add(component);
```

```
        }

        public void Remove(IFinanceComponent component)
        {
            _children.Remove(component);
        }

        public List<IFinanceComponent> GetChildren()
        {
            return _children;
        }
    }
}
```

Листовий компонент (Models/Fund.cs):

```
using personalAccounting.Patterns.Composite;
using System.Collections.Generic;

namespace personalAccounting.Models
{
    public class Fund : IFinanceComponent
    {
        public int FundId { get; set; }
        public string FundName { get; set; }
        public decimal Balance { get; set; }
        public int UserId { get; set; }

        public decimal GetTotalBalance()
        {
```

```
            return Balance;
        }

        public void Add(IFinanceComponent component) { }
        public void Remove(IFinanceComponent component) { }
        public List<IFinanceComponent> GetChildren()
        {
            return new List<IFinanceComponent>();
        }
    }
}
```

Використання у Фасаді (Patterns/Facade/AccountingFacade.cs):

```
        public decimal GetTotalCapital(int userId)
        {
            var userAssets = new UserAssets { UserName = $"Assets for User {userId}" };

            var accounts = _accountRepository.GetAllByUserId(userId);
            foreach (var acc in accounts)
            {
                userAssets.Add(acc);
            }

            var funds = _fundRepository.GetAllByUserId(userId);
            foreach (var fund in funds)
            {
                userAssets.Add(fund);
            }
```

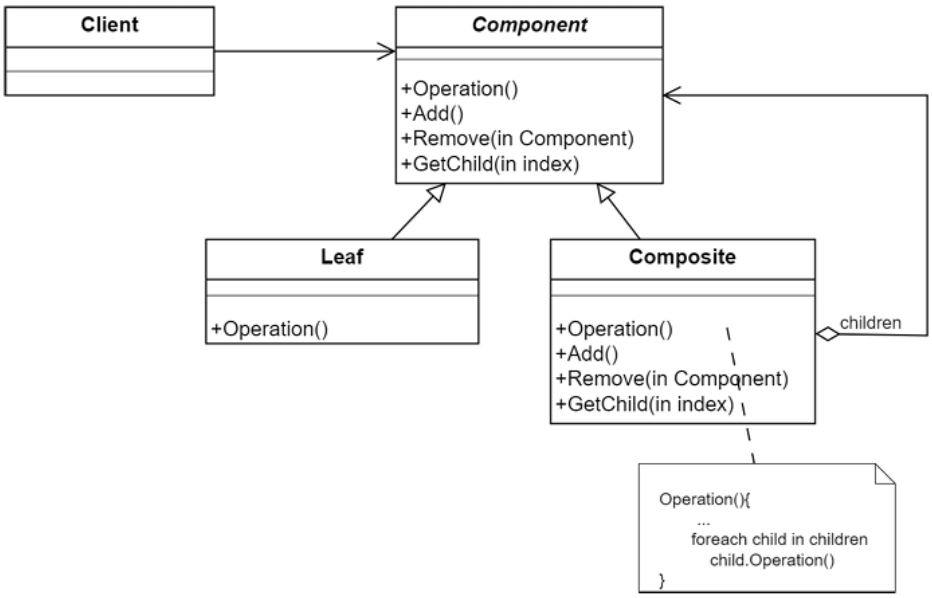
```
        return userAssets.GetTotalBalance();
    }
```

Контрольні питання

1. Яке призначення шаблону «Композит»?

Дозволяє будувати деревоподібні структури об'єктів (ієрархія «частина-ціле»). Головна перевага в тому, що клієнт може взаємодіяти як з простими елементами, так і зі складними групами елементів однаково — через єдиний інтерфейс.

2. Нарисуйте структуру шаблону «Композит».



3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

Component: спільний інтерфейс для всіх елементів дерева.

Leaf: кінцевий елемент («листок»), що виконує основну роботу.

Composite: контейнер («гілка»), що зберігає дочірні компоненти.

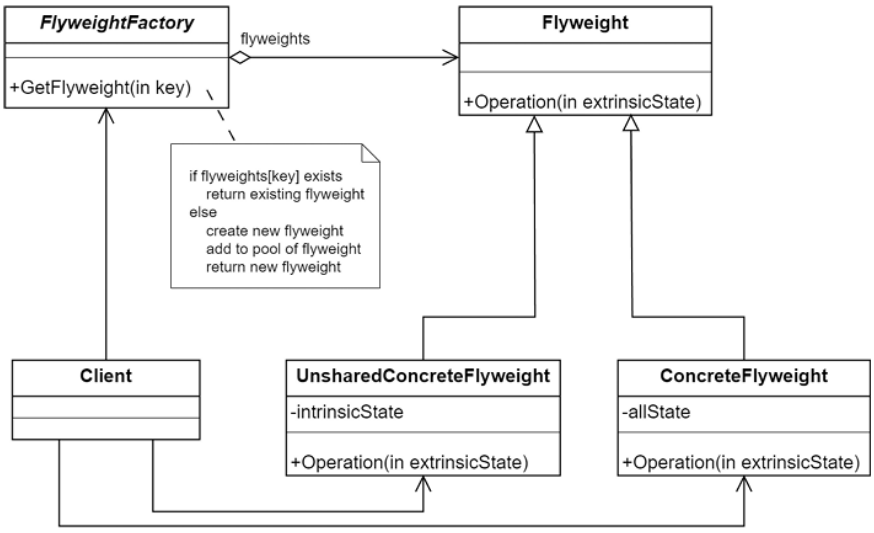
Client: працює з деревом через Component.

Взаємодія: Клієнт надсилає запит. Якщо це Leaf — він виконує дію. Якщо Composite — він ретранслює запит усім своїм піделементам, можливо, додаючи проміжну обробку.

4. Яке призначення шаблону «Легковаговик»?

Цей патерн призначений для економії оперативної пам'яті при роботі з величезною кількістю об'єктів. Він виокремлює спільний (незмінний) стан об'єктів і зберігає його в одному екземплярі, який використовується багатьма об'єктами одночасно, замість дублювання даних.

5. Нарисуйте структуру шаблону «Легковаговик».



6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

Flyweight: інтерфейс.

ConcreteFlyweight: зберігає спільний (внутрішній) стан.

FlyweightFactory: керує пулом (кешем) легковаговиків.

Взаємодія: Клієнт запитує об'єкт у Фабрики. Фабрика повертає вже існуючий екземпляр або створює новий. Унікальні дані (зовнішній стан) клієнт передає в методи Легковаговика як параметри безпосередньо при виклику.

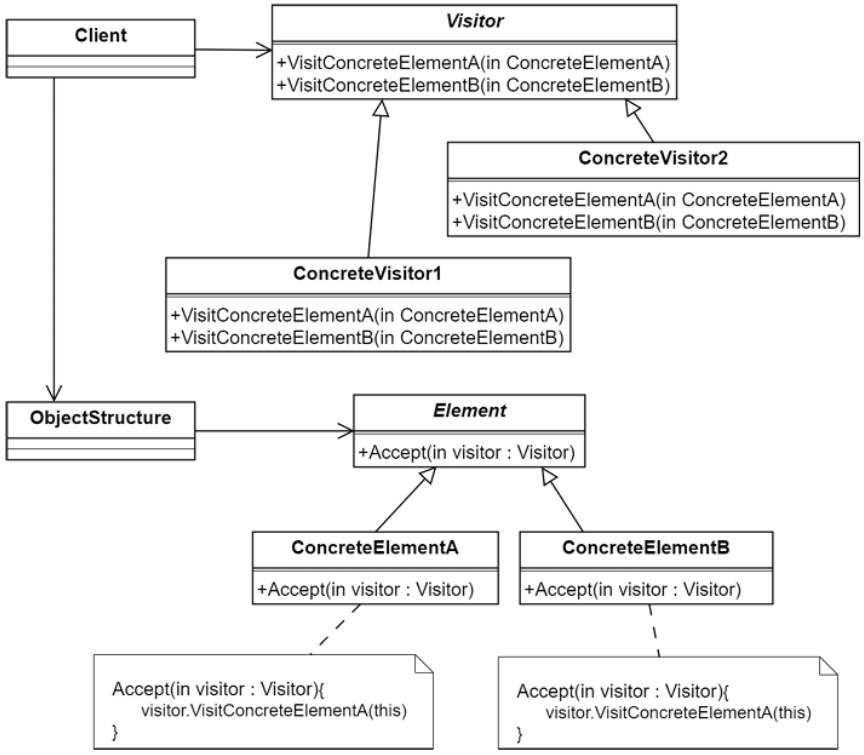
7. Яке призначення шаблону «Інтерпретатор»?

Використовується для визначення граматики простої мови та створення інтерпретатора, який аналізує та виконує речення цієї мови. Часто застосовується для розбору математичних виразів, командних рядків або регулярних виразів.

8. Яке призначення шаблону «Відвідувач»?

Дозволяє додавати нову функціональність (операції) до ієрархії класів без зміни коду самих класів. Алгоритм виконання операції виноситься в окремий клас-відвідувач.

9. Нарисуйте структуру шаблону «Відвідувач».



10. Які класи входять в шаблон «Відвідувач», та яка між ними взаємодія?

Visitor: інтерфейс, що описує методи відвідування для кожного типу елемента.

ConcreteVisitor: реалізація конкретної операції.

Element: інтерфейс із методом accept.

Взаємодія: Працює через подвійну диспетчеризацію. Клієнт викликає метод accept(visitor) у Елемента. Елемент "впускає" відвідувача, викликаючи його метод visit(this) і передаючи посилання на себе. Так Відвідувач отримує доступ до даних Елемента і виконує потрібну логіку.

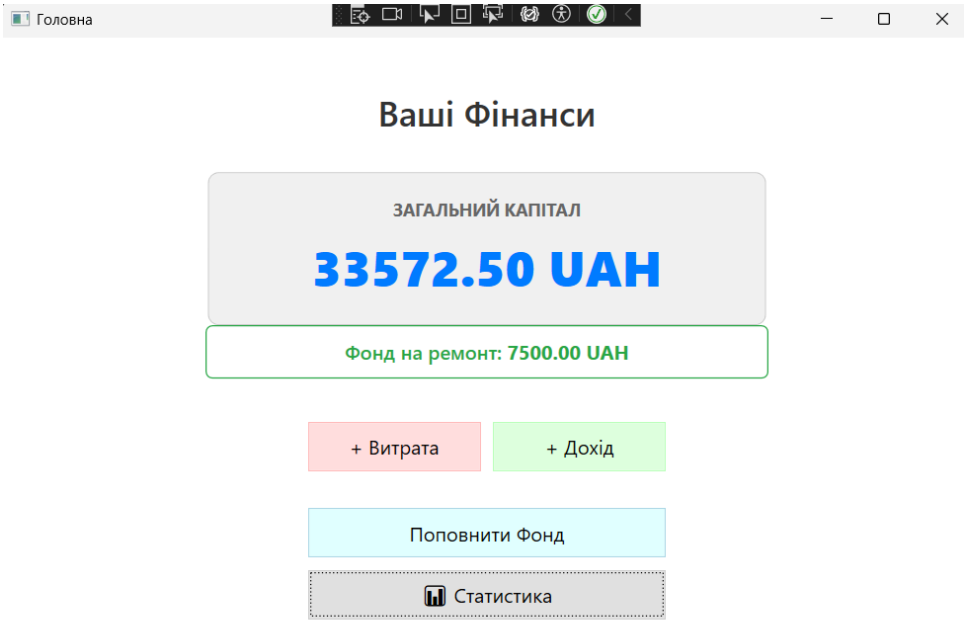
Висновки: Під час виконання лабораторної роботи я ознайомилась із принципами роботи та структурою структурних і поведінкових шаблонів проєктування, зокрема «Composite» (Компонувальник).
Практична частина роботи полягала в реалізації структурного шаблону Composite (Компонувальник). Було створено інтерфейс IFinanceComponent та його реалізації: Account і Fund (як листи) та UserAssets (як контейнер). Це дозволило об'єднати різні фінансові активи користувача в єдину ієрархічну структуру. Впровадження Компонувальника дало змогу розрахувати загальний капітал (суму всіх рахунків і фондів) за допомогою єдиного виклику методу GetTotalBalance() на контейнері UserAssets, що значно спростило бізнес-логіку у клієнтському коді та наочно продемонструвало коректний облік коштів між основним рахунком та цільовим фондом.

Репозиторій: [посилання]

ДОДАТКИ

Додаток А

Головне вікно програми з відображенням загального капіталу, розрахованого за допомогою патерну Composite (сума рахунків та фондів).



Додаток Б

Вікно статистики, що відображає історію транзакцій, включаючи автоматично створену операцію переказу коштів у цільовий фонд.