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

Лабораторна робота №8
Технології розроблення програмного забезпечення
«Патерни проектування»

Тема: Особиста бухгалтерія

Виконала:
студентка групи ІА-32
Коляда М. С.

Перевірила:
Мягкий М.Ю.

Київ 2025

Зміст

Завдання..... 3

Теоретичні відомості..... 3

Діаграма класів:.....4

Опис реалізації патерну«Flyweight»..... 4

Фрагменти програмного коду:..... 6

Вихідний код:..... 9

Питання до лабораторної роботи..... 9

Висновок.....12

Тема: Патерни проектування.

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.

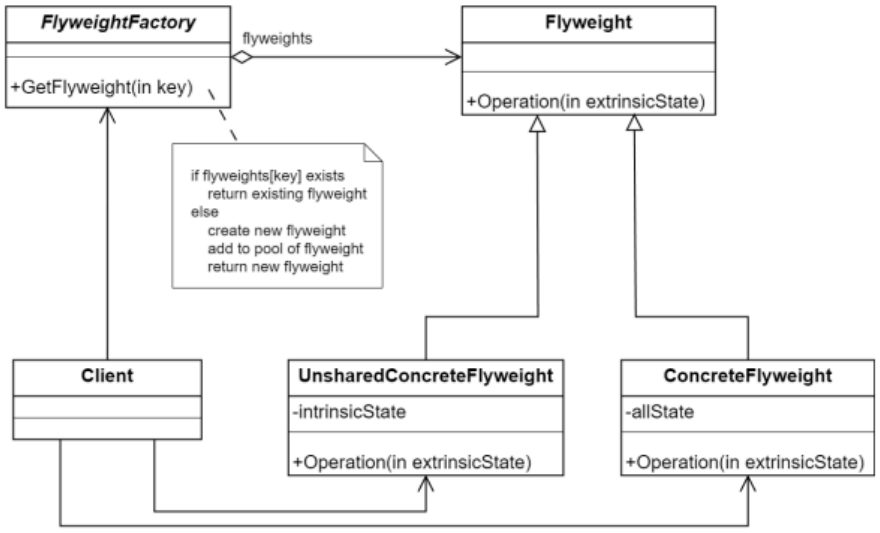
• Реалізувати не менше 3-х класів відповідно до обраної теми.

• 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

Теоретичні відомості

Шаблон «Flyweight»

Призначення: Шаблон використовується для зменшення кількості об'єктів в додатку шляхом поділу цих об'єктів між ділянками додатку. Flyweight являє собою поділюваний об'єкт. Дуже важливою є концепція «внутрішнього» і «зовнішнього» станів. Внутрішній стан відображає дані, характерні саме поділюваному об'єкту (наприклад, код букви); зовнішній стан несе інформацію про його застосування в додатку (наприклад, рядок і стовпчик). Внутрішній стан зберігається в самому поділюваному об'єкті, зовнішній – в об'єктах додатку (контексту використання поділюваного об'єкта).



Переваги та недоліки:

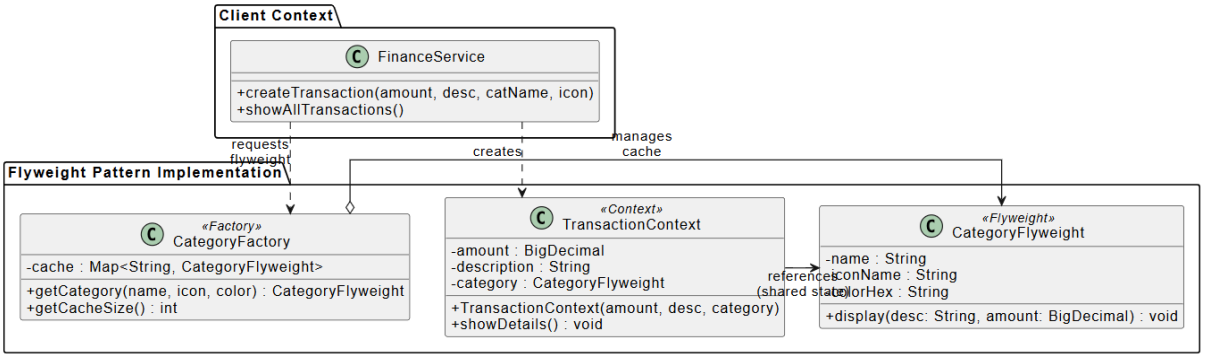
+ Заощаджує оперативну пам’ять.

- Витрачає процесорний час на пошук/обчислення контексту.

- Ускладнює код програми внаслідок введення безлічі додаткових класів.

Хід роботи

Діаграма класів:



Опис реалізації патерну«Flyweight»

Реалізовано структурний патерн Flyweight (Легковаговик) для оптимізації використання оперативної пам'яті в системі «Особиста бухгалтерія». Метою впровадження патерну була необхідність уникнути дублювання даних при створенні великої кількості об'єктів транзакцій, які мають

повторювані атрибути (наприклад, однакові категорії витрат, такі як «Їжа» або «Транспорт»).

Система розділяє стан об'єкта на дві частини:

1. Внутрішній стан (Intrinsic State): Спільні дані, що не змінюються (назва категорії, іконка, колір). Вони зберігаються в об'єкті-легковаговику.
2. Зовнішній стан (Extrinsic State): Унікальні дані конкретної транзакції (сума, опис, дата). Вони зберігаються в контексті.

Механізм роботи: Клас CategoryFactory керує кешем (пулом) об'єктів CategoryFlyweight. Коли клієнт (FinanceService) запитує категорію, фабрика перевіряє її наявність у Map. Якщо об'єкт існує – повертається посилання на нього, якщо ні – створюється новий. Таким чином, тисячі транзакцій можуть посилатися на один фізичний екземпляр категорії в пам'яті.

Учасники патерну:

- CategoryFactory (Factory / Фабрика) Клас, що відповідає за створення та управління легковаговиками. Він містить колекцію cache (відображення Map<String, CategoryFlyweight>) для зберігання вже створених категорій. Метод getCategory() гарантує, що для однакових параметрів категорії буде повернуто той самий об'єкт, забезпечуючи принцип спільного використання (sharing).
- CategoryFlyweight (Flyweight / Легковаговик) Клас, що містить спільний (внутрішній) стан: name, iconName, colorHex. Ці дані ініціалізуються один раз і не можуть бути змінені ззовні. Клас також містить метод display(), який приймає зовнішній стан (суму та опис) як аргументи для виконання операцій.
- TransactionContext (Context / Контекст) Клас, що представляє конкретну транзакцію. Він зберігає унікальний (зовнішній) стан: amount (сума) та description (опис). Також він містить посилання на

об'єкт CategoryFlyweight. Цей клас об'єднує спільні та унікальні дані для повноцінної роботи з транзакцією.

- FinanceService (Client Context / Клієнт) Клас бізнес-логіки, який ініціює створення транзакцій. Він звертається до фабрики для отримання легковаговика та створює контексти транзакцій.

Зв'язки: Фабрика (CategoryFactory) агрегує (містить) Легковаговиків (CategoryFlyweight). Контекст (TransactionContext) посилається на Легковаговика. Клієнт (FinanceService) використовує Фабрику для управління життєвим циклом спільних об'єктів.

Обґрунтування вибору шаблону «Flyweight»

Для вирішення задачі зберігання транзакцій було обрано саме шаблон Flyweight:

- У системі планується зберігання десятків тисяч записів. Без використання патерну кожна транзакція створювала б власний об'єкт Category з дубльованими рядками (назви, шляхи до іконок). Flyweight дозволяє зменшити кількість об'єктів категорій з тисяч до кількох десятків (за кількістю унікальних категорій).
- Зменшення кількості об'єктів знижує навантаження на Garbage Collector (збирач сміття) Java, що покращує загальну швидкодію програми при роботі з великими списками даних.

Фрагменти програмного коду:

CategoryFlyweight.java

```
package ia32.koliada.finance.flyweight;

import java.math.BigDecimal;

public class CategoryFlyweight {
    private String name;
```

```
    private String iconName;
    private String colorHex;

    public CategoryFlyweight(String name, String iconName, String colorHex) {
        this.name = name;
        this.iconName = iconName;
        this.colorHex = colorHex;
    }

        public void display(String description, BigDecimal amount) {
        System.out.printf("Відображення: [Категорія: %s | Іконка: %s] ->
Транзакція: '%s' (%s грн)%n",
                name, iconName, description, amount);
    }

    @Override
    public String toString() {
        return name;
    }
}
```

CategoryFactory.java

```
package ia32.koliada.finance.flyweight;

import java.util.HashMap;
import java.util.Map;

public class CategoryFactory {
        private static final Map<String, CategoryFlyweight> cache = new
HashMap<>();

    public static CategoryFlyweight getCategory(String name, String iconName,
String colorHex) {
              if (!cache.containsKey(name)) {
            System.out.println(">>> [Factory] Створення нового об'єкта категорії:
" + name);
            cache.put(name, new CategoryFlyweight(name, iconName, colorHex));
```

```
        } else {
            System.out.println("... [Factory] Використання існуючого об'єкта: " +
name);
        }
        return cache.get(name);
    }

    public static int getCacheSize() {
        return cache.size();
    }
}
```

TransactionContext.java

```
package ia32.koliada.finance.flyweight;

import java.math.BigDecimal;

public class TransactionContext {
    private BigDecimal amount;
    private String description;

     private CategoryFlyweight category;

    public TransactionContext(BigDecimal amount, String description, String
catName, String icon, String color) {
        this.amount = amount;
        this.description = description;
            this.category = CategoryFactory.getCategory(catName, icon, color);
    }

    public void showDetails() {
             category.display(description, amount);
    }
}
```

Вихідний код:

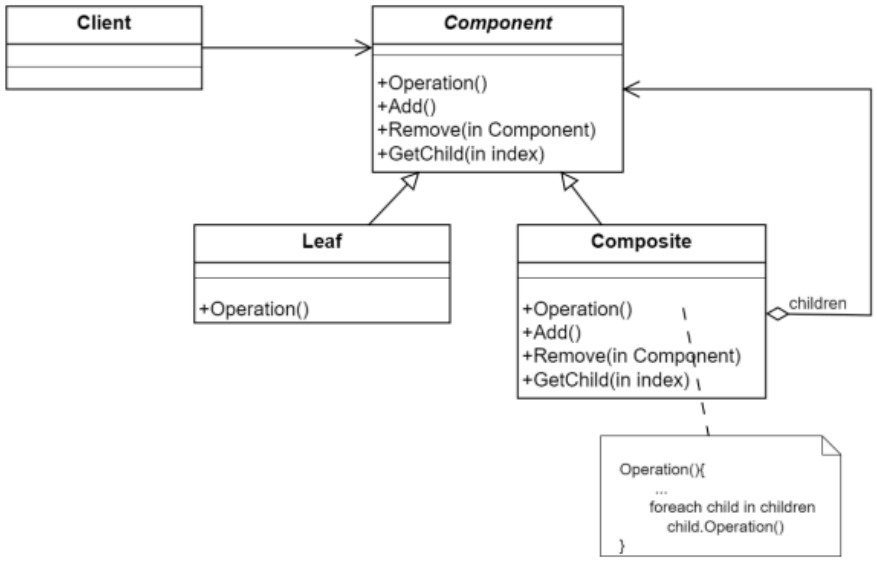
https://github.com/mariiakoliada/TRPZ/tree/master/src/ia32/koliada/finance/flyweight

Питання до лабораторної роботи

1. Яке призначення шаблону «Композит»?

Шаблон дозволяє згрупувати об'єкти в деревоподібні структури для представлення ієрархії «частина-ціле» та дозволяє клієнтам працювати з окремими об'єктами та їх групами однаково (через спільний інтерфейс).

2. Нарисуйте структуру шаблону «Композит».



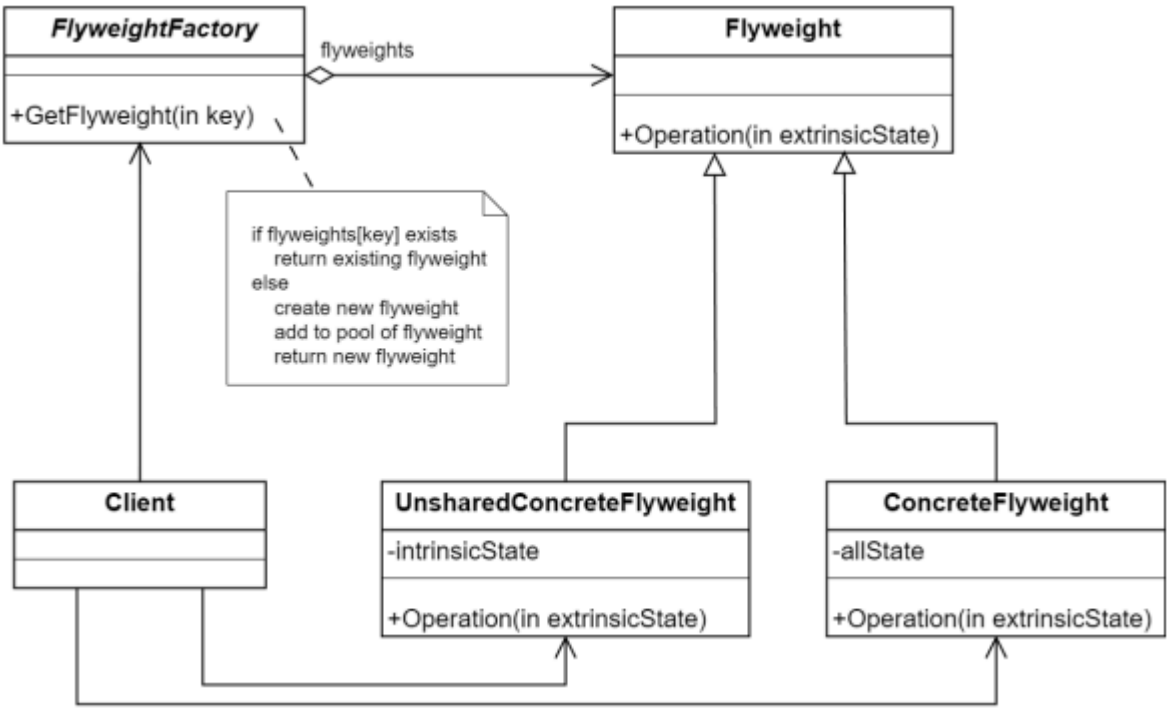
3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

До складу входять: Component (спільний інтерфейс), Leaf (простий елемент, що виконує роботу), Composite (контейнер для дочірніх елементів) та Client. Взаємодія: Клієнт посилає запит через інтерфейс Component. Якщо отримувач — Leaf, він виконує дію. Якщо отримувач — Composite, він перенаправляє запит усім своїм дочірнім елементам і може додавати власну логіку.

4. Яке призначення шаблону «Легковаговик»?

Шаблон дозволяє ефективно підтримувати велику кількість дрібних об'єктів шляхом спільного використання їхнього внутрішнього (незмінного) стану замість зберігання однакових даних у кожному об'єкті окремо.

5. Нарисуйте структуру шаблону «Легковаговик».



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

До складу входять: Flyweight (інтерфейс), ConcreteFlyweight (зберігає внутрішній стан), FlyweightFactory (створює та керує пулом об'єктів), Client. Взаємодія: Клієнт звертається до Фабрики за легковаговиком. Фабрика повертає існуючий екземпляр або створює новий. Клієнт викликає методи легковаговика, передаючи йому зовнішній (унікальний) стан як аргумент.

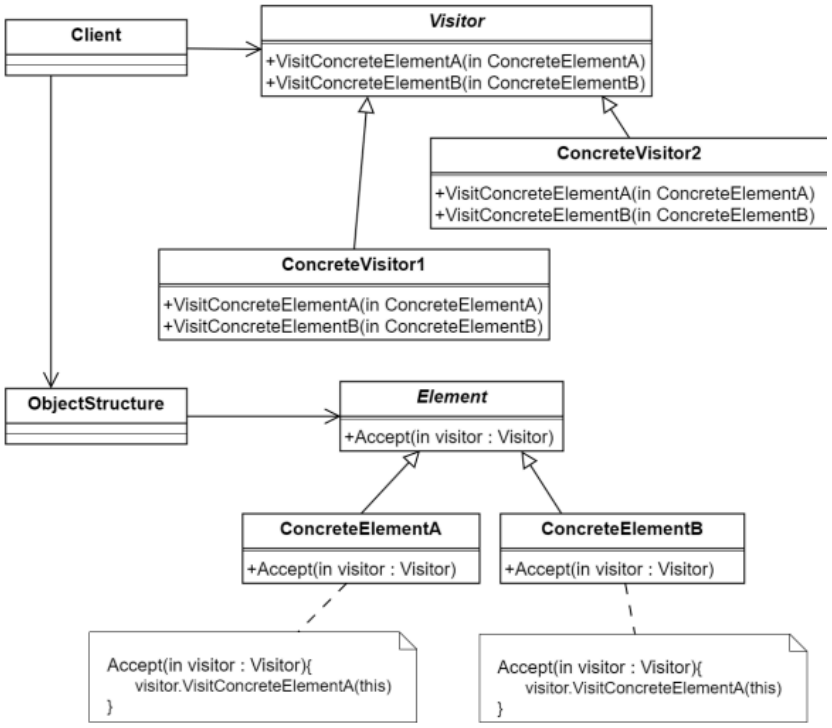
7. Яке призначення шаблону «Інтерпретатор»?

Шаблон визначає представлення граматики для заданої мови та інтерпретатор, який використовує це представлення для аналізу та виконання речень цієї мови (часто використовується для парсингу виразів).

8. Яке призначення шаблону «Відвідувач»?

Шаблон дозволяє додавати нові операції до об'єктів, не змінюючи класів цих об'єктів. Він відокремлює алгоритм від структури об'єкта, над яким цей алгоритм оперує.

9. Нарисуйте структуру шаблону «Відвідувач».



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

До складу входять: Visitor (інтерфейс операцій), ConcreteVisitor (реалізація операцій), Element (інтерфейс з методом accept), ConcreteElement.

Взаємодія: Клієнт проходить по елементах структури. Елемент у методі accept викликає відповідний метод Відвідувача (наприклад, visitElementA), передаючи посилання на себе (this). Таким чином Відвідувач отримує доступ до даних Елемента і виконує потрібну дію.

Висновок: під час виконання лабораторної роботи, було вивчено структуру та особливості застосування структурних патернів проєктування. На прикладі системи «Особиста бухгалтерія» успішно

реалізовано шаблон «Flyweight» (Легковаговик). Застосування цього патерну дозволило значно оптимізувати роботу з пам'яттю програми за рахунок відокремлення повторюваних даних (атрибутів категорій) від унікальних даних транзакцій.