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

із дисципліни *«Технології розроблення програмного забезпечення»*

Тема: “ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»”

Виконала:
Студентка групи ІА-24
Сіденко Д.Д.

Перевірив:
Мягкий М.Ю.

Київ-2024

Тема лабораторних робіт:

Варіант 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.).

Завдання:

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

Зміст

Хід роботи:

Крок 1: Короткі теоретичні відомості

Патерн Composite дозволяє об'єднувати об'єкти в деревоподібну структуру для представлення частин та цілих, що дає можливість працювати з окремими об'єктами та їх групами однаково. Цей патерн застосовується для створення ієрархічних структур, таких як файлові системи чи графи.

Патерн Flyweight дозволяє економити пам'ять, розділяючи об'єкти, які мають спільні властивості, при цьому створюється одна інстанція об'єкта для спільних частин і використовуються окремі об'єкти для специфічних властивостей. Це корисно, коли потрібно створювати багато подібних об'єктів, наприклад, пікселі на екрані або символи в тексті.

Патерн Interpreter визначає граматику мови та її інтерпретацію, створюючи інтерпретатор для виразів або мов за допомогою абстракцій, які розбивають синтаксис на компоненти. Зазвичай він використовується для побудови інтерпретаторів специфічних мов, таких як SQL чи математичні вирази.

Патерн Visitor дозволяє додавати нову поведінку об'єктам без зміни їх класів, створюючи окремий об'єкт-візитер, який відвідує інші об'єкти та виконує операції на них. Це дозволяє додавати нові операції до об'єктів без зміни їх структури, наприклад, для обробки різних типів елементів у складі колекції.

Крок 2: Реалізація шаблону “Flyweight”

Шаблон Flyweight — це структурний шаблон проектування, який використовується для зменшення використання пам'яті шляхом спільного використання об'єктів, які мають однакові значення або властивості. Шаблон дозволяє ефективно працювати з великими об'ємами даних, коли багато об'єктів мають однакові або схожі стани, таким чином зменшуючи кількість необхідних об'єктів у пам'яті.

У системі що реалізує особисту бухгалтерію я вирішила використати даний шаблон для оптимізації використання об'єктів типу транзакції, зменшуючи витрати пам'яті за рахунок спільного використання екземплярів об'єктів з однаковими характеристиками. Ось як це вийшло:

Клас TransactionTypeFlyweight - це об'єкт, що представляє тип транзакції (наприклад, "дохід" або "витрата"). Клас має приватне поле description, яке містить опис типу транзакції. Шаблон Flyweight застосовується до об'єктів типів транзакцій, оскільки багато транзакцій можуть мати однаковий опис (тип).

Статичний клас Factory має кеш (мапу) для зберігання вже створених екземплярів TransactionTypeFlyweight.

Коли запитується об'єкт з певним описом транзакції, фабрика перевіряє, чи вже є такий об'єкт у кеші. Якщо немає — створює новий і додає його в кеш.

Якщо об'єкт з таким описом вже існує в кеші, повертається його екземпляр без створення нового.

Клас Transaction - це сутність транзакції, яка містить всі дані, такі як сума, опис, дата та тип транзакції.

Для зберігання типу транзакції в об'єкті використовується поле transactionTypeDescription (тип опису, наприклад, "дохід" або "витрата").

Метод setTransactionType встановлює значення для цього опису.

Метод getTransactionType повертає екземпляр TransactionTypeFlyweight, що відповідає опису типу транзакції, за допомогою фабрики.

```java
public class TransactionTypeFlyweight {  5 usages  new *
    private String description;  2 usages

    public TransactionTypeFlyweight(String description) {  1 usage  new *
        this.description = description;
    }

    public String getDescription() {  no usages  new *
        return description;
    }

    public static class Factory {  1 usage  new *
        private static final Map<String, TransactionTypeFlyweight> cache = new HashMap<>();  3 usages

        public static TransactionTypeFlyweight getTransactionType(String description) {  1 usage  new *
            if (!cache.containsKey(description)) {
                cache.put(description, new TransactionTypeFlyweight(description));
            }
            return cache.get(description);
        }
    }
}
```

Рис. 1 - Клас TransactionTypeFlyweight

```java
public class Transaction {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String description;

    @Column(nullable = false)
    private Double amount;

    @Column(nullable = false)
    private boolean isPeriodic;

    @Column(nullable = false)
    private LocalDate transactionDate;

    @ManyToOne
    @JoinColumn(name = "account_id")
    private Account account;

    @Column(nullable = false)
    private String transactionTypeDescription;

    public void setTransactionType(String description) {  1 usage  new *
        this.transactionTypeDescription = description;
    }
    public TransactionTypeFlyweight getTransactionType() {  no usages  new *
        return TransactionTypeFlyweight.Factory.getTransactionType(this.transactionTypeDescription);
    }
}
```

Рис. 1 - Клас Transaction

Проєкт можна переглянути у репозиторії:
https://github.com/papiroskada/BudgetMate

Висновок: у рамках виконання лабораторної роботи щодо проектування системи "Особиста бухгалтерія" було здійснено реалізацію шаблону проектування Flyweight для оптимізувації використання об'єктів типу транзакції. Для цього спочатку було проаналізовано функціональні вимоги системи, а потім створено необхідні класи та інтерфейси. Також завдяки даній лабораторній роботі я була ознайомлена з такими шаблонами проектування, як: «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR».