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 №7

із дисципліни *«Технології розроблення програмного забезпечення»*

Тема: “Шаблони «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»”

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

Короткі теоретичні відомості 3

Реалізація шаблону проектування “Bridge” 4

Висновок 5

Хід роботи:

Крок 1: Короткі теоретичні відомості

Шаблон “Mediator” дозволяє об'єктам взаємодіяти один з одним без необхідності знати про конкретних партнерів. Замість того, щоб об'єкти напряму зверталися один до одного, їх взаємодія керується медіатором, що централізує логіку взаємодії.

Шаблон “Facade” надає спрощений інтерфейс для складної підсистеми. Він приховує складність і робить взаємодію з підсистемою більш зручною та зрозумілою для користувача.

Шаблон “Bridge” дозволяє розділити абстракцію та її реалізацію, щоб їх можна було змінювати незалежно. Він дає змогу змінювати реалізацію без необхідності змінювати код абстракції.

Шаблон “Template Method” визначає структуру алгоритму, але дозволяє підкласам реалізовувати деякі кроки алгоритму. Це дозволяє зберегти загальний алгоритм, змінюючи лише окремі його частини.

Крок 2: Реалізація шаблону “Bridge”

Шаблон Bridge використовується для поділу абстракції і її реалізації таким чином, щоб їх можна було змінювати незалежно. Це дозволяє змінювати реалізацію абстракції без змін у самій абстракції, що дає гнучкість і полегшує підтримку коду.

Структуру шаблону "Bridge" можна виразити наступним чином:

Абстракція (Abstraction): Це клас, що містить посилання на реалізацію (підклас «Implementation»). Абстракція використовує це посилання для делегування операцій до конкретної реалізації.

Розширена Абстракція (RefinedAbstraction): Це підклас абстракції, який додає додаткові функціональні можливості до базового класу абстракції, використовуючи реалізацію через делегування.

Реалізація (Implementation): Це інтерфейс, що визначає операції, які повинні бути реалізовані. Абстракція делегує конкретним класам реалізації виконання операцій.

Конкретна реалізація (ConcreteImplementation): Це конкретний клас, який реалізує інтерфейс «Implementation». Він надає конкретну реалізацію операцій.

Я вирішила використати даний шаблон при програмуванні роботи із отриманням статистики та генерації файлів з нею. Відповідно було розроблено абстрактний клас StatisticsFileHandler, що реалізує методи, але не визначає, як саме генерувати ці файли.

```
public abstract class StatisticsFileHandler{  9 usages  2 inheritors  dashushak

    private static final String FOLDER = "statistics_file/";  1 usage

    protected static final String FILE_PATH = "statistics_file/%s.%s";  2 usages

    protected static final List<String> TRANSACTION_TABLE_COLUMNS_NAME =  2 usages
            List.of("№", "Sum", "Refill", "Comment", "Category", "Date and time");

    protected static final List<String> ACCUMULATIONS_TABLE_COLUMNS_NAME =  2 usages
            List.of("№", "Name", "Comment", "Current sum", "Goal sum", "Start date", "End date",
                    "Last payment date", "Status");

    protected static final List<String> FINANCIAL_ARRANGEMENT_TABLE_COLUMNS_NAME =  2 usages
            List.of( ...elements: "№", "Name", "Type", "Percent", "StartSum", "Current Sum",
                    "Refund Sum", "Start date", "End date", "From to user funds", "Status");


    public abstract File generateStatisticsFile(List<Transaction> transactions, List<Accumulation> accumulations,  1 usage
                                List<FinancialArrangement> financialArrangements, String userEmail);

    public static void deleteAllFile() {...}
}
```

Рис. 1 - абстрактний клас StatisticsFileHandler

Далі класи ExcelStatisticsFileHandler та WordStatisticsFileHandler, що надають конкретні реалізації для генерації статистики у форматах Excel та Word відповідно. Вони розширюють абстракцію StatisticsFileHandler, надаючи конкретну реалізацію для кожного формату файлів. Тобто, вони забезпечують конкретну логіку створення файлів в залежності від формату. Окрім того вони також містять специфічні деталі реалізації, такі як робота з бібліотеками для створення Excel (Apache POI) або Word (Apache POI) файлів. Кожен з цих класів має свою конкретну реалізацію для роботи з відповідними форматами.

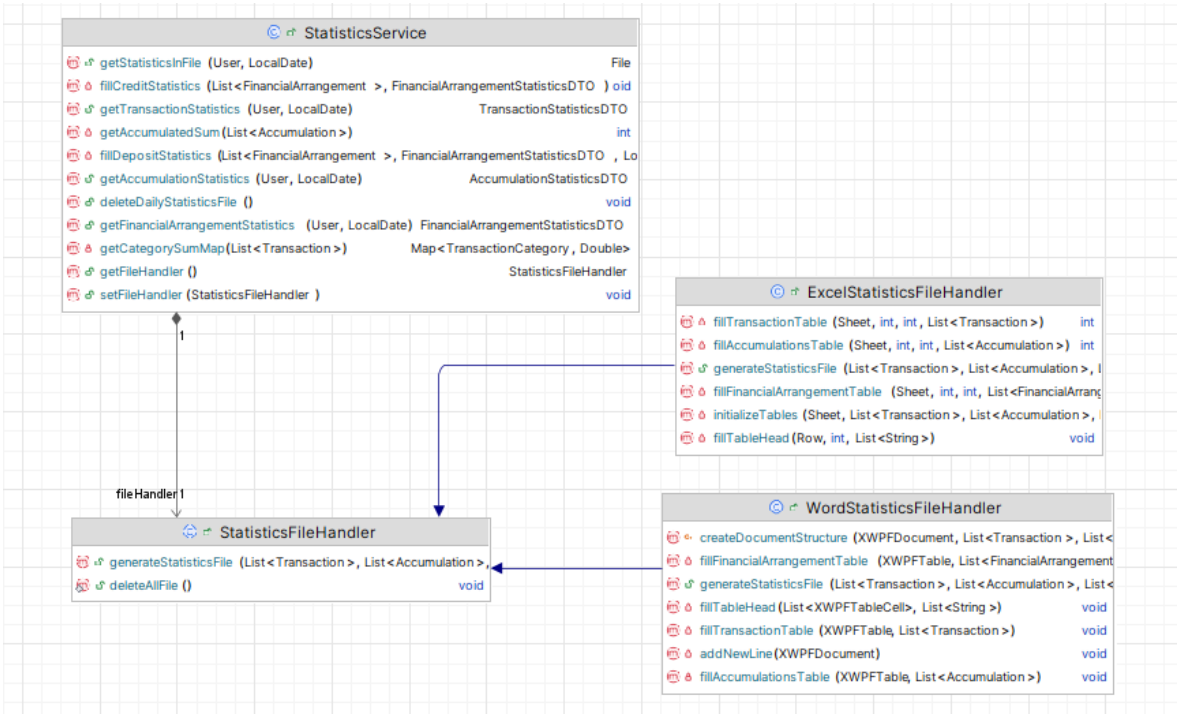

Рис.2 - Діаграма класів, що реалізують шаблон “Bridge”

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 Decorator для створення валідації транзакцій. 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: «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator».