Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №7

З дисципліни “Технології розроблення програмного забезпечення”

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

Виконав
студент групи ІА–22:
Прохоров О.Д.

Перевірив
Мягкий М. Ю.

Київ 2024

Зміст

Тема: ШАБЛОНИ «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»

Мета: Навчитися використанню одного з зазначених шаблонів проектування.

Хід роботи:

Короткі теоретичні відомості:

Шаблон «Facade» (фасад) передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми. Оскільки підсистема може складатися з безлічі класів, а кількість її функцій - не більше десяти, то щоб уникнути створення «спагеті-коду» (коли все тісно пов'язано між собою) виділяють один загальний інтерфейс доступу, здатний правильним чином звертатися до внутрішніх деталей.

Фасад - це простий інтерфейс для роботи зі складною підсистемою, що містить безліч класів. Фасад може мати урізаний інтерфейс, який не має 100% функціональності, якої можна досягти, використовуючи складну підсистему безпосередньо. Але він надає саме ті фічі, які потрібні клієнту, і приховує всі інші.

Фасад корисний, якщо ви використовуєте якусь складну бібліотеку з безліччю рухомих частин, але вам потрібна тільки частина її можливостей.

Крок 1. Код шаблону Facade.

Для виконання цієї лабораторної роботи були написані наступні нові класи:

SystemFacade.java, UserService.java, Receipt.java та змінено код головного файлу проекту. На наступних рисунках(1.1, 1.2, 1.3, 2.1) відображено код цих класів.

```java
3 usages
public class SystemFacade extends UserService {


    1 usage
    public void createAndDistributeShoppingList(List<User> users, List<Goods> goods, DistributionStrategy strategy) {
        strategy.distribute(goods, users);
        System.out.println("Shopping list distributed using " + strategy.getClass().getSimpleName());
    }

    1 usage
    public void attachReceiptToGood(List<User> users, Long userId, Long goodId, Receipt receipt) {
        User user = findUserById(users, userId);
        if (user != null) {
            Goods good = user.getShoppingLists().stream() Stream<Goods>
                    .filter(g -> g.getId().equals(goodId))
                    .findFirst() Optional<Goods>
                    .orElse( other: null);

            if (good != null) {
                good.setReceipt(receipt);
                good.setActPrice(receipt.getAmount());
                System.out.println("Receipt attached to " + good.getName());
            } else {
                System.out.println("Good not found for user " + user.getName());
            }
        } else {
            System.out.println("User not found with ID " + userId);
        }
    }
}
```

Рисунок 1.1 SystemFacade.java

```java
4 usages  1 inheritor
public class UserService {
    1 usage
    public List<User> getUsers() {
        return List.of(
                new User( Id: 1L,  name: "Alice",  email: "alice@example.com"),
                new User( Id: 2L,  name: "Bob",  email: "bob@example.com")
        );
    }
    public List<Goods> getGoods() {
        return List.of(
                new Goods( id: 1L,  name: "Laptop",  description: "Electronics",  estPrice: 1200.0,  actPrice: 1200.0),
                new Goods( id: 2L,  name: "Phone",  description: "Electronics",  estPrice: 800.0,  actPrice: 800.0)
        );
    }
    1 usage
    public User findUserById(List<User> users, Long userId) {
        return users.stream() Stream<User>
                .filter(user -> user.getId().equals(userId))
                .findFirst() Optional<User>
                .orElse( other: null);
    }
}
```

Рисунок 1.2 UserService.java

```java
8 usages
public class Receipt {
    4 usages
    private Long id;
    4 usages
    private double amount;
    4 usages
    private LocalDate date;
    4 usages
    private String description;
    no usages
    public Receipt() {
    }

    1 usage
    public Receipt(Long id, double amount, LocalDate date, String description) {
        this.id = id;
        this.amount = amount;
        this.date = date;
        this.description = description;
    }
    no usages
    public Long getId() { return id; }

    no usages
    public void setId(Long id) { this.id = id; }

    1 usage
    public double getAmount() {
        return amount;
    }

    no usages
    public void setAmount(double amount) { this.amount = amount; }

    no usages
    public LocalDate getDate() { return date; }

    no usages
    public void setDate(LocalDate date) { this.date = date; }
```

Рисунок 1.3 Receipt.java

Крок 2. Основний класс системи.

```java
@SpringBootApplication
public class CpsProjectConsoleApplication {
        public static void main(String[] args) {
            SpringApplication.run(CpsProjectConsoleApplication.class, args);
                SystemFacade facade = new SystemFacade();
                UserService userService = new UserService();

                List<User> users = userService.getUsers();
                List<Goods> goods = userService.getGoods();
                Receipt receipt = new Receipt( id: 1L,  amount: 750.0, LocalDate.now(),  description: "Payment for Phone");


                facade.createAndDistributeShoppingList(users, goods, new EvenDistributionStrategy());
                facade.attachReceiptToGood(users,  userId: 2L,  goodId: 2L, receipt);
                users.forEach(user -> {
                    System.out.println(user.getName() + "'s shopping list: " + user.getShoppingLists());
                });
            }
}
```

Рисунок 2.1 CpsProjectConsoleApplication.java

Крок 3. Результати програми

На рисунку 3.1 можна побачити, що програма успішно скомпілювала код та видала необхідний результат, яким є:

- вивід використаної стратегії
- додавання чеку до товару
- виведення списку товарів всіх користувачів

```
Shopping list distributed using EvenDistributionStrategy
Receipt attached to Phone
Alice's shopping list: [Goods{id=1, name='Laptop', desc='Electronics', Estimated price=1200.0', Actual price=1200.0', receipt=No receipt}]
Bob's shopping list: [Goods{id=2, name='Phone', desc='Electronics', Estimated price=800.0', Actual price=750.0', receipt=Receipt{id=1, amount=750.0, date=2024-12-21, description='Payment for Phone'}}]
```

Рисунок 3.1 Results

Висновки.

В результаті цієї лабораторної роботи було виконано дії з вивчення зазначених патернів проектування, в результаті якого я визначив, що для нашої системи найбільше підходить саме патерн Facade, створення коду, що відповідає шаблону проектування Facade та базового коду проекта. Результати програми було виведено через консоль для демонстрації роботи програми.