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 №4

із дисципліни *«Технології розроблення програмного забезпечення»*

Тема: «ШАБЛОНИ “SINGLETON”, “ITERATOR”, “PROXY”, “STATE”, “STRATEGY”»

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

3. Застосування одного з розглянутих шаблонів при реалізації програми.

Зміст

Короткі теоретичні відомості 3

Реалізація шаблону проектування “State” 4

Висновок 6

Хід роботи:

Крок 1: Короткі теоретичні відомості

Singleton: Цей шаблон гарантує, що клас має лише один екземпляр, і надає глобальну точку доступу до нього. Використовується, коли потрібен єдиний об'єкт для управління ресурсами, наприклад, підключенням до бази даних.

Iterator: Шаблон надає послідовний доступ до елементів колекції без розкриття її внутрішньої структури. Зручний для обходу об'єктів, таких як списки або масиви.

Proxy: Шаблон надає об'єкт-заступник для контролю доступу до іншого об'єкта. Proxy може виконувати додаткові дії, наприклад, перевірку прав доступу або відкладену ініціалізацію ресурсу.

State: Дозволяє об'єкту змінювати свою поведінку в залежності від поточного стану. Зовні виглядає, що об'єкт змінює свій клас. Використовується, коли поведінка об'єкта залежить від його стану.

Strategy: Визначає сімейство алгоритмів, інкапсулює їх і робить взаємозамінними. Це дозволяє динамічно обирати алгоритм залежно від умов, наприклад, різні методи сортування даних.

Крок 2: Реалізація шаблону “State”

Для реалізації шаблону “State” перш за все потрібно проаналізувати які класи потребують створення декількох станів. Для цього варто згадати які класи будуть головними моделями системи та який функціонал закладений у технічному завданні. Серед головних класів виділяємо наступні: User, Account, Fund, Loan, Transaction, Receipt, Statistics. Для створення різних рахунків; ведення єдиного фонду на всі рахунки (всією сім'єю); можливість введення вкладів / кредитів для контролю банківських рахунків (звірка нарахованих відсотків з необхідними і т.д.) потрібно реалізувати варіанти різних рахунків та їх станів. Так, наприклад, будь який рахунок має мати такі стани:

- Активний;
- Заблокований;
- Закритий.

Так як моделі кредиту та фонду зберігають за собою подібний функціонал до звичайного рахунку, було прийнято рішення створити батьківський клас Account, та наслідувати класи Loan та Fund.

Реалізація шаблону “State” вимагає створення окремих класів для реалізації певної поведінки в залежності від стану об’єкту.

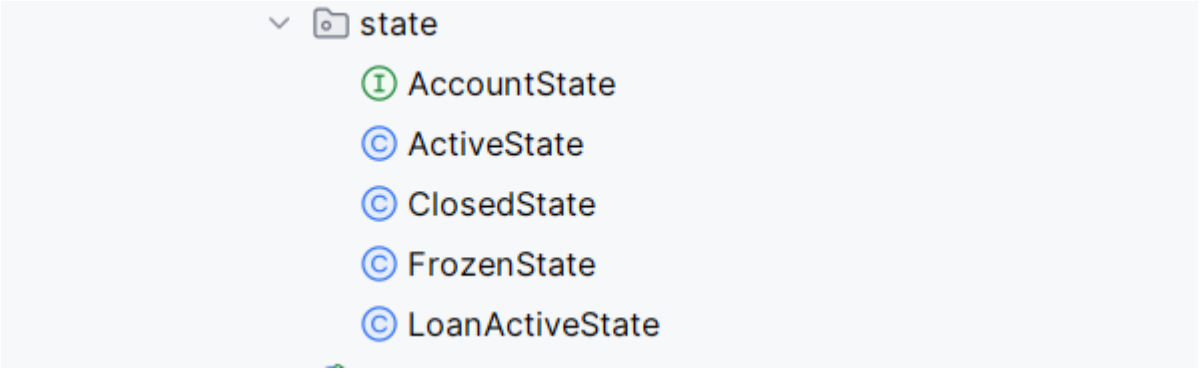

Рис. 1 - класи, що реалізують стани об’єктів

Опис створених об’єктів:

Інтерфейс AccountState:

Це загальний інтерфейс, що визначає методи для взаємодії з рахунком незалежно від його стану. Це такі методи як updateBalance, deposit, withdraw, resetBalance, які дозволяють змінювати баланс рахунку в залежності від його стану.

Класи станів (наприклад, ActiveState, FrozenState, ClosedState, LoanActiveState):

Ці класи реалізують інтерфейс AccountState і визначають конкретну поведінку для кожного стану.

Кожен клас змінює поведінку рахунку (чи кредиту) для відповідного стану. Наприклад:

ActiveState: Рахунок активний, тому можна здійснювати депозити та зняття коштів.

FrozenState: Рахунок заморожений, тому жодні операції не дозволені.

ClosedState: Рахунок закритий, тому з ним не можна здійснювати операції.

LoanActiveState: Для кредиту є специфічні умови щодо внесення платежів, відповідно є потреба в створенні додаткового класу з реалізацією функціоналу.

Приклад реалізації шаблону в класі головного об’єкту “Account” (рахунок):

AccountState.java

```
1  package com.budget.budgetmate.state;
2
3  import com.budget.budgetmate.models.Account;
4
5  public interface AccountState {  5 usages  4 implementations
6      void updateBalance(Account account, double amount)
7      void deposit(Account account, double amount);  1 usa
8      void withdraw(Account account, double amount);  1 us
9      void resetBalance(Account account);  1 usage  3 implem
10 }
11
```

Account.java | Fund.java | Loan.java | Receipt.java | Transaction.java

```
14 public class Account {
31
32     public Account() {  dashushak
33         this.state = new ActiveState();
34     }
35
36     public void updateBalance(double amount) {  2 usages  2 overrides  dashushak
37         state.updateBalance( account: this, amount);
38     }
39
40     public void deposit(double amount) {  no usages  dashushak
41         state.deposit( account: this, amount);
42     }
43     public void withdraw(double amount) {  no usages  dashushak
44         state.withdraw( account: this, amount);
45     }
46     public void resetBalance() {  no usages  dashushak
47         state.resetBalance( account: this);
48     }
49     public void activateAccount() {  no usages  dashushak
50         this.setState(new ActiveState());
51     }
52     public void freezeAccount() {  no usages  dashushak
53         this.setState(new FrozenState());
54     }
55     public void closeAccount() {  1 usage  dashushak
56         this.setState(new ClosedState());
57     }
58
59 }
```

Рис. 2 - Клас Account та інтерфейс Account State

Проєкт можна переглянути у репозиторії:
https://github.com/papiroskada/budget-mate

Висновок: у рамках виконання лабораторної роботи щодо проектування системи "Особиста бухгалтерія" було здійснено реалізацію шаблону проектування State для класів Account, Loan, Fund. 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: “SINGLETON”, “ITERATOR”, “PROXY”, “STATE”, “STRATEGY”.