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
Технологія розроблення програмного забезпечення

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

«Застосування шаблону Template Method в темі Download Manager»

Варіант 26

Виконала
студентка групи ІА-13
Танасієнко Анастасія Вячеславівна

Київ 2023р.

Мета: дослідження шаблонів «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD» реалізація шаблону проектування «TEMPLATE METHOD».

Шаблону проектування «MEDIATOR»

Шаблон проектування "Mediator" відноситься до патернів поведінки і використовується для визначення способу організації комунікації між об'єктами без прямих посилань між ними. Цей шаблон рекомендує використовувати посередника (Mediator), який взаємодіє з об'єктами і керує комунікацією між ними. Такий підхід дозволяє розділити об'єкти системи і зменшити залежність між ними.

Основні учасники шаблону:

Mediator (Посередник): Визначає інтерфейс для комунікації між об'єктами.

ConcreteMediator (Конкретний посередник): Реалізує інтерфейс посередника і забезпечує координацію між об'єктами.

Colleague (Колега): Визначає інтерфейс для взаємодії з посередником та знаходження інших колег.

ConcreteColleague (Конкретний колега): Реалізує інтерфейс колеги і взаємодіє з іншими колегами через посередника.

Принцип роботи:

Колеги взаємодіють лише з посередником, а не напряму один з одним.

Посередник координує комунікацію між колегами.

Зміни в одному колеги можуть впливати на інших, але це відбувається через посередника.

Шаблону проектування «FACADE»

Шаблон проектування "FACADE" (фасад) відноситься до структурних патернів і надає уніфікований інтерфейс до підсистеми великої кількості класів, інтерфейс яких не завжди є зручним для використання. Фасад покращує читабельність, спрощує взаємодію із складною системою, надаючи один спрощений інтерфейс.

Основні елементи шаблону "FACADE":

Facade (Фасад): Надає простий і зручний інтерфейс для взаємодії зі складною системою. Цей клас знає, як взаємодіяти з усіма необхідними класами або підсистемами.

Subsystem Classes (Класи підсистем): Система може складатися з багатьох класів, які можуть бути досить складними для використання окремо.

Принцип роботи шаблону полягає в тому, що клієнт взаємодіє лише з фасадом, а не з усіма класами підсистеми напряму.

Шаблону проектування «BRIDGE»

Шаблон проектування "Bridge" відноситься до структурних патернів і дозволяє розділити абстракцію від її реалізації так, щоб обидві могли змінюватися незалежно одна від одної. Цей шаблон використовує композицію замість наслідування для досягнення мети.

Основні учасники шаблону "Bridge":

Abstraction (Абстракція): Визначає базовий інтерфейс абстракції та утримує посилання на об'єкт реалізації.

RefinedAbstraction (Вдосконалена абстракція): Розширює інтерфейс абстракції.

Implementor (Реалізатор): Визначає інтерфейс для класу реалізації.

ConcreteImplementor (Конкретний реалізатор): Реалізує інтерфейс реалізації.

Принцип роботи шаблону полягає в тому, що абстракція і реалізація можуть змінюватися незалежно одна від одної. Це дозволяє мати різні варіанти абстракцій та реалізацій і з'єднувати їх динамічно під час виконання програми.

Шаблону проектування «TEMPLATE METHOD»

Шаблон проектування "Template Method" відноситься до поведінкових патернів і дозволяє визначити скелет алгоритму в базовому класі, але визначити деякі кроки алгоритму у підкласах. Цей шаблон дозволяє підкласам перевизначати певні частини алгоритму без зміни його структури.

Основні учасники шаблону "Template Method":

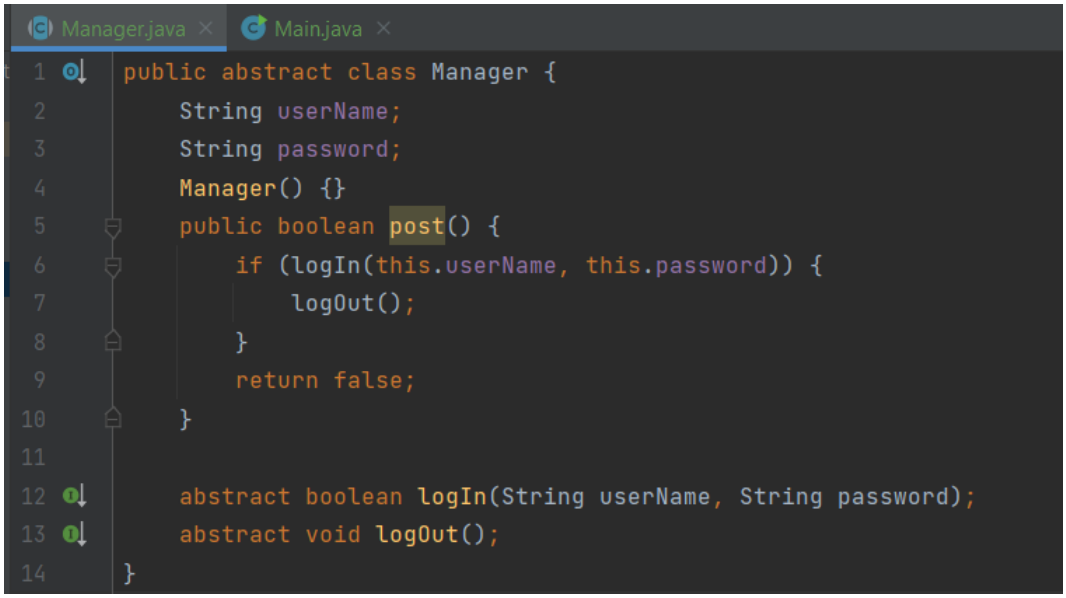
AbstractClass (Абстрактний клас): Визначає скелет алгоритму у вигляді методів, але залишає деякі кроки алгоритму для конкретних підкласів.

ConcreteClass (Конкретний клас): Реалізує конкретні кроки алгоритму, які визначені у базовому класі.

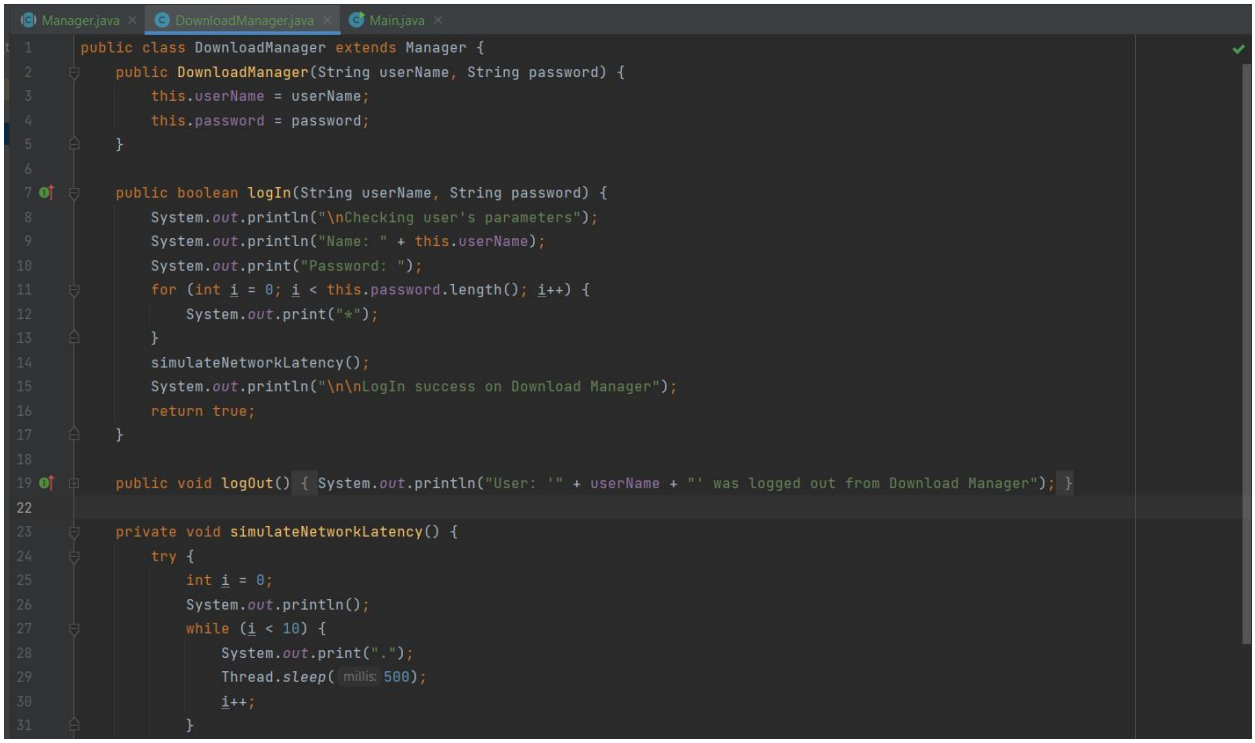
Принцип роботи шаблону полягає в тому, що базовий клас визначає загальний алгоритм, а підкласи можуть перевизначати окремі його частини за необхідності.

Дослідження шаблону

Цей код представляє абстрактний клас Manager з полями userName та password. Метод post() викликає методи logIn() і logOut(). Метод logIn() та logOut() є абстрактними і повинні бути реалізовані в підкласах.



Цей код представляє клас DownloadManager, який є підкласом абстрактного класу Manager. У конструкторі передаються ім'я користувача та пароль. Методи logIn() та logOut() реалізовані відповідно для входу та виходу з менеджера завантажень. Є також приватний метод simulateNetworkLatency(), який імітує затримку мережі.



```
            }
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}

```

Це консольний застосунок на Java для введення імені та паролю користувача. Створюється об'єкт DownloadManager, передаються введені дані, і викликається метод post() для взаємодії з менеджером завантажень.

Manager.java | DownloadManager.java | Main.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {

        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        Manager manager = null;
        System.out.print("Input user name: ");
        String userName = reader.readLine();
        System.out.print("Input password: ");
        String password = reader.readLine();
        manager = new DownloadManager(userName, password);
        manager.post();
    }
}

```

Висновок:

У лабораторній роботі були вивчені та реалізовані шаблони проектування "MEDIATOR", "FACADE", "BRIDGE" та "TEMPLATE METHOD".

"MEDIATOR" дозволяє розділити систему на компоненти, які взаємодіють через посередника.

"FACADE" забезпечує простий інтерфейс для взаємодії зі складною системою.

"BRIDGE" дозволяє відокремити абстракцію від реалізації, полегшуючи розширення системи.

"TEMPLATE METHOD" визначає загальний алгоритм у базовому класі, дозволяючи підкласам перевизначати окремі його частини.

У реалізації "TEMPLATE METHOD" було показано, як створити скелет алгоритму у базовому класі, а підкласи можуть реалізовувати власні частини алгоритму.

Загальна висновок: Дослідження та використання цих шаблонів дозволяє покращити структуру програмного забезпечення, зменшити залежності та забезпечити гнучкість у розширенні функціоналу.