МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Навчально-науковий інститут комп’ютерного моделювання, прикладної фізики та математики

Звіт

З лабораторної роботи №4

З дисципліни:“ Ефективність та якість архітектурних рішень інформаційних систем ”

На тему:

«Патерн проєктування Адаптер»

Виконав:

студент групи ІКМ-М225В

Загорулько Віталій Олегович

Харків 2025

Мета роботи

Метою даної лабораторної роботи є ґрунтовне вивчення та практичне засвоєння структурного патерна проєктування Адаптер (Adapter), а також набуття навичок його застосування для інтеграції існуючих класів із несумісними інтерфейсами в існуючу програмну систему, не змінюючи її базову структуру.

Завдання

У межах виконання лабораторної роботи необхідно виконати такі завдання:

- Ознайомитися з теоретичними засадами та основними ідеями патерна проєктування Адаптер.
- Використовуючи наданий інтерфейс Notification, який не можна змінювати, інтегрувати дві нові, несумісні системи сповіщень: Slack та SMS.
- Створити структуру класів, що демонструє реалізацію патерна Адаптер для забезпечення сумісності.
- Навести приклад клієнтського коду, що демонструє відправку сповіщень через усі три канали (Email, Slack, SMS).

Теоретичні відомості та Опис Проєкту

Патерн Адаптер (Adapter) — це ключовий структурний патерн проєктування, який розв'язує проблему несумісності інтерфейсів між об'єктами. Його основна ідея полягає у створенні посередника, який перетворює інтерфейс одного класу (Adaptee) на інтерфейс, очікуваний клієнтом (Target). У контексті розробки архітектурних рішень, Адаптер є незамінним інструментом для інтеграції зовнішніх бібліотек, API або застарілих компонентів у сучасну систему, не порушуючи її стабільність та принципи. Структура патерна передбачає наявність чотирьох основних елементів: Target (цільовий інтерфейс, у нашому випадку Notification), Adaptee (класи, які потрібно інтегрувати — SlackApi та SmsSender), Adapter (посередник, який реалізує Target) та Client (клієнтський код).

У межах даної лабораторної роботи було поставлено завдання інтегрувати два нові канали сповіщення — Slack та SMS — у систему, яка вже використовує незмінний інтерфейс Target (Notification). Цей інтерфейс вимагає єдиного методу send(String title, String message). Однак, зовнішні сервіси Adaptee мають несумісні вимоги: SlackApi вимагає спеціальні дані для авторизації (login, apiKey, chatId) і метод, який може приймати лише один текстовий рядок, а SmsSender вимагає дані відправника (phone, sender) та часто обмежене тіло повідомлення. Для вирішення цієї проблеми були розроблені два конкретні класи-адаптери: SlackNotificationAdapter та SmsNotificationAdapter. Кожен із цих класів реалізує цільовий інтерфейс Notification, тим самим задовольняючи вимоги клієнтського коду. Усередині адаптера відбувається інтеграційна логіка: він отримує виклик send(title, message) і перетворює його на специфічний виклик Adaptee. Наприклад, SlackNotificationAdapter об'єднує заголовок і повідомлення в один рядок перед викликом методу SlackApi::post(), тоді як SmsNotificationAdapter виконує обрізання тексту для дотримання лімітів SMS. Таким чином, усі деталі авторизації та специфіка API інкапсулюються в адаптерах. У клієнтському коді демонструється, що об'єкти EmailNotification та обидва об'єкти-адаптери використовуються єдинотипно, взаємодіючи лише з Notification, що підкреслює ефективність даного архітектурного рішення.

Програмний код реалізації на Java розміщено у Додатку А.

Висновок

Під час виконання лабораторної роботи було успішно вивчено та реалізовано структурний патерн Адаптер. Головна мета — інтеграція нових, несумісних сервісів (Slack та SMS) у жорстко заданий інтерфейс Notification — була досягнута. Класи-адаптери виступили як посередники, що "перекладають" стандартний виклик send(title, message) на специфічні методи Adaptee. Це дозволило зберегти незмінним існуючий цільовий інтерфейс, забезпечити гнучкість архітектури, дозволяючи додавати нові, несумісні функціональності, та уніфікувати взаємодію клієнтського коду з різними каналами сповіщення. Отримані знання підтверджують ефективність використання патерна Адаптер для вирішення проблем сумісності, що є критично важливим для створення підтримуваних та розширюваних програмних систем.

Додаток А

Lab4.java