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 №9

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

На тему:

«Патерн проєктування Відвідувач (Visitor)»

Виконав:

студент групи ІКМ-М225В

Загорулько Віталій Олегович

Харків 2025

Мета роботи

Метою даної лабораторної роботи є ґрунтовне вивчення та практичне засвоєння поведінкового патерна проєктування Відвідувач (Visitor), а також набуття навичок його застосування для відокремлення алгоритму від об'єктів, над якими він працює, що дозволяє додавати нові операції без зміни класів об'єктної структури.

Завдання

У межах виконання лабораторної роботи необхідно виконати такі завдання:

1. Ознайомитися з теоретичними засадами та основними ідеями патерна проєктування Відвідувач.
2. Створити структуру об'єктів (Елементи): Company, Department, Employee.
3. Організувати можливість отримання репорту "Зарплатна відомість" (Salary Report) для всієї компанії та для конкретного департаменту.
4. Реалізувати цю можливість за допомогою патерна Відвідувач, де репорт є конкретним Відвідувачем.
5. Створити структуру класів та методів, що вирішує описане завдання, та приклад клієнтського коду, що демонструє вибірковий запит репорту.

Теоретичні відомості та Опис Проєкту

Патерн Відвідувач (Visitor) — це поведінковий патерн проєктування, який дозволяє додавати нові операції до існуючих ієрархій класів Елементів, не змінюючи самі класи цих елементів . Патерн особливо корисний, коли необхідно виконати схожу, але специфічну для кожного класу операцію над групою об'єктів, що ідеально підходить для обробки складних об'єктних структур, таких як організаційні дерева чи компілятори. Це реалізується шляхом механізму подвійної диспетчеризації (double dispatch).

У межах цієї лабораторної роботи було реалізовано можливість генерації звіту "Зарплатна відомість" для організаційної структури, що складається з Елементів (Company, Department, Employee). Кожен із цих елементів реалізує інтерфейс OrganizationElement, який вимагає наявності єдиного методу accept(OrganizationVisitor visitor).

1. Ієрархія Елементів (Організаційна структура): Класи Company, Department та Employee представляють незмінну структуру даних. У методі accept() кожен елемент викликає відповідний метод visit() об'єкта Відвідувача, передаючи себе як параметр. Наприклад, Company.accept(visitor) викликає visitor.visitCompany(this).
2. Ієрархія Відвідувачів (Операція): Інтерфейс OrganizationVisitor оголошує специфічні методи для кожного типу елемента (visitCompany, visitDepartment, visitEmployee). Клас SalaryReportVisitor (Конкретний Відвідувач) реалізує ці методи, інкапсулюючи всю логіку збору та форматування даних про зарплату. Ця логіка включає рекурсивний обхід структури: visitCompany ініціює visitDepartment, який, у свою чергу, ініціює visitEmployee.

Таким чином, уся бізнес-логіка створення звіту про зарплату відокремлена від класів організаційної структури. Це дозволяє легко додати нові типи звітів (наприклад, "Звіт про кількість персоналу" або "Звіт про податки") шляхом створення нового класу Відвідувача, без необхідності модифікувати Company, Department чи Employee. Клієнтський код демонструє гнучкість, викликаючи

операцію accept() як на об'єкті всієї компанії, так і на окремому об'єкті департаменту.

Програмний код реалізації на Java розміщено у Додатку А.

Висновок

Під час виконання лабораторної роботи було успішно вивчено та реалізовано поведінковий патерн Відвідувач (Visitor). Завдання додавання нової операції "отримання зарплатної відомості" до ієрархії організаційних сутностей було вирішено шляхом створення окремого класу SalaryReportVisitor.

Ключовим результатом є те, що класи Елементів залишилися незмінними, а весь алгоритм репорту інкапсульовано в класі Відвідувача. Це підтверджує ефективність патерна Відвідувач як інструменту для відокремлення операції від об'єктної структури, що забезпечує гнучкість, легке додавання нових операцій і дотримання принципу відкритості/закритості (OCP).

Додаток А

Lab9.java

-