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 №8

Технологія розробки програмного забезпечення

Шаблони «Composite», «Flyweight», «Interpreter», «Visitor»

Виконала студентка
групи ІА-23:
Кашуб’як С. М.

Перевірив:
Мягкий М. Ю.

Київ 2024

Тема: Шаблони «Composite», «Flyweight», «Interpreter», «Visitor».

Мета: Ознайомитися з принципами роботи шаблонів проектування «Composite», «Flyweight», «Interpreter», «Visitor» їх перевагами та недоліками. Набути практичних навичок у застосуванні шаблону adapter при розробці програмного забезпечення на прикладі реалізації архіватора.

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

Варіант:

..14 Архіватор (strategy, adapter, factory method, facade, visitor, p2p)

Архіватор повинен являти собою візуальний додаток з можливістю створення і редагування архівів різного типу (.tar.gz, .zip, .rar, .ace) додавання/ видалення файлів / папок, редагування метаданих (по можливості), перевірка checksum архівів, тестування архівів на наявність пошкоджень, розбиття архівів на частини.

ЗМІСТ

Теоретичні відомості

Visitor – це поведінковий шаблон проектування, який дозволяє визначати нові операції над об’єктами без зміни їхньої структури. Шаблон відокремлює алгоритм від об’єктів, над якими він виконується.

Ключові компоненти

1. Visitor: інтерфейс або абстрактний клас, який визначає методи відвідування для кожного типу елементів.
2. ConcreteVisitor: конкретна реалізація відвідувача, яка виконує певну операцію над елементами.
3. Element: інтерфейс або абстрактний клас, який визначає метод accept для прийняття відвідувача.
4. ConcreteElement: конкретні реалізації елементів, які викликають відповідні методи відвідувача через accept.
5. ObjectStructure: клас, який зберігає колекцію елементів та дозволяє відвідувачу обійти всі ці елементи.

Шаблон Visitor використовується, коли:

- потрібно виконувати різні операції над об’єктами, але не змінювати їхню структуру.
- необхідно уникнути додавання методів до класів, оскільки це може порушити принципи інкапсуляції.
- логіка операцій повинна бути відокремлена від логіки зберігання даних.

Переваги

1. Розширюваність: легко додавати нові операції, створюючи нові класи-відвідувачі.
2. Розділення відповідальностей: логіка операцій винесена у відвідувача, залишаючи елементи простими.
3. Гнучкість: дозволяє виконувати різні операції над елементами без змін у їхній структурі.

Недоліки

1. Труднощі з додаванням нових елементів: якщо додається новий тип елемента, доводиться змінювати всі існуючі відвідувачі.
2. Порушення інкапсуляції: відвідувач може отримувати доступ до внутрішнього стану елементів, що може бути небажаним.

Хід Роботи

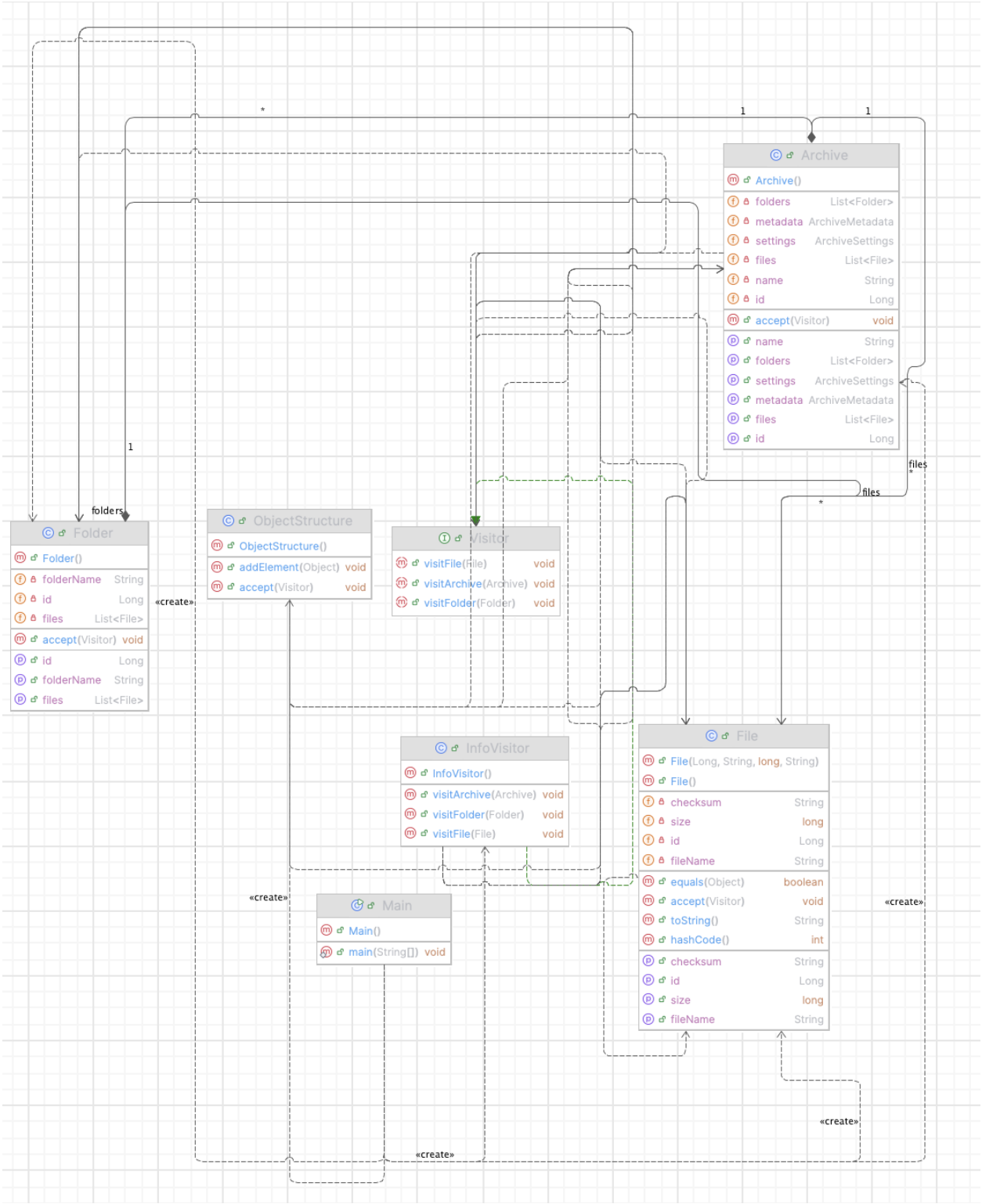

Рис 1. Діаграма класів

Діаграма класів

Діаграма демонструє реалізацію шаблону Visitor у проєкті, який дозволяє виконувати операції над елементами архівної системи без зміни їхньої структури. Visitor спрощує додавання нових операцій, надаючи можливість групувати їх у спеціальних класах-відвідувачах.

Основні елементи діаграми:

VisitorPattern

Visitor – це інтерфейс, який визначає методи для відвідування елементів системи:

- visitFile – обробляє елемент типу File.
- visitFolder – обробляє елемент типу Folder.
- visitArchive – обробляє елемент типу Archive.

Конкретний відвідувач

InfoVisitor – реалізує інтерфейс Visitor та виконує конкретну операцію: збирає інформацію про елементи системи:

- visitFile – виводить інформацію про файл (назва, розмір тощо)
- visitFolder – виводить інформацію про папку (назва, кількість файлів тощо).
- visitArchive – виводить інформацію про архів (назва, вміст тощо).

Елементи (Concrete Elements)

Усі елементи системи реалізують метод accept, який приймає об’єкт Visitor та викликає відповідний метод відвідування.

File:

- має властивості: id, fileName, size, checksum.
- реалізує метод accept, який викликає visitFile у відвідувача.

Folder:

- має властивості: id, folderName, список файлів (files).
- реалізує метод accept, який викликає visitFolder у відвідувача.

Archive:

- має властивості: id, name, список файлів (files), список папок (folders), налаштування (settings), метадані (metadata).
- реалізує метод accept, який викликає visitArchive у відвідувача.

ObjectStructure

ObjectStructure – клас, який керує колекцією елементів та забезпечує їх ітерацію:

- addElement – додає елемент до списку.
- accept – викликає метод accept для кожного елемента у колекції, передаючи йому відвідувача.

Використання Visitor

Клієнтський код взаємодіє з об’єктами системи через Visitor:

- створює об’єкти File, Folder, Archive.
- додає їх до ObjectStructure.
- викликає метод accept для ObjectStructure, передаючи йому об’єкт Visitor.

```
--- Visitor Pattern Demonstration ---
File: example.txt, Size: 1234
File: example2.txt, Size: 5678
Folder: MyFolder, Files: 2
Archive: MyArchive
```

Рис 2. Застосування шаблону при реалізації програми

Робота патерну

Перевірка файлу

Роль шаблону Visitor: Visitor обробляє елемент File, викликаючи метод visitFile у класі InfoVisitor, щоб отримати детальну інформацію про файл (назва, розмір, контрольна сума).

Спрощення: логіка обробки файлу зосереджена в одному місці – у класі InfoVisitor, що дозволяє уникнути дублювання коду та полегшує додавання нових операцій.

Перевірка папки

Роль шаблону Visitor: Visitor викликає метод visitFolder у класі InfoVisitor, щоб отримати інформацію про папку, включаючи її назву та кількість файлів.

Спрощення: шаблон дозволяє централізувати логіку для папок, залишаючи клас Folder простим і орієнтованим на зберігання даних.

Перевірка архіву

Роль шаблону Visitor: Visitor викликає метод visitArchive у класі InfoVisitor, щоб зібрати інформацію про архів (назва, вкладені файли та папки).

Спрощення: логіка для роботи з архівами винесена в клас InfoVisitor, що робить клас Archive менш завантаженим і простим у підтримці.

Клієнт викликає accept для структури ObjectStructure. ObjectStructure ітерує по своїх елементах і викликає їхні методи accept. Кожен елемент викликає відповідний метод visitXXX у переданого відвідувача (Visitor). Конкретний відвідувач (InfoVisitor) реалізує специфічну операцію над кожним типом елемента (File, Folder, Archive).

Архітектурне спрощення

- Розподіл відповідальностей: шаблон “Visitor” інкапсулює логіку операцій у відвідувачах, залишаючи елементи (File, Folder, Archive) лише відповідальними за свої дані.

- Гнучкість: додавання нових операцій (наприклад, перевірки цілісності чи підрахунку загального розміру) зводиться до створення нового відвідувача, без змін у класах елементів.

- Інкапсуляція: клієнтський код взаємодіє лише з ObjectStructure і Visitor, не знаючи деталей реалізації операцій.

Переваги використання шаблону Visitor у цьому випадку

1. Централізація логіки обробки

Шаблон “Visitor” дозволяє централізувати логіку операцій над елементами системи:

- уся логіка роботи з файлами, папками та архівами зосереджена у класі відвідувача (InfoVisitor).
- елементи (File, Folder, Archive) залишаються простими, відповідаючи лише за зберігання даних та виклик методу accept.

Результат: зменшується дублювання коду та спрощується підтримка проекту.

2. Розширюваність

Завдяки шаблону “Visitor”, можна легко додавати нові операції над елементами без змін у самих елементах. Наприклад, додавання нового відвідувача SizeCalculatorVisitor для підрахунку загального розміру архіву, папки чи файлу не вимагає змін у класах File, Folder, чи Archive.

Результат: зменшується ризик порушення існуючого функціоналу при розширенні системи.

3. Уніфікація доступу

Шаблон “Visitor” забезпечує єдиний інтерфейс для виконання операцій над різними типами елементів:

- кожен елемент реалізує метод accept, який передає себе відвідувачу.
- це дозволяє клієнтському коду працювати однаково з усіма елементами, незалежно від їх типу.

Результат: клієнтський код залишається простим і не залежить від деталей реалізації об’єктів.

4. Інкапсуляція змін

Шаблон “Visitor” дозволяє ізолювати зміни, пов’язані з обробкою елементів, у класах відвідувачів: наприклад, якщо зміниться алгоритм збору інформації про файли, це вплине лише на клас InfoVisitor, а не на File чи інші елементи.

Результат: знижується зв’язаність компонентів, що робить проект більш гнучким.

5. Легке тестування

Операції, інкапсульовані у відвідувачах, можна тестувати окремо від класів елементів: наприклад, тестування InfoVisitor не потребує модифікацій класів File, Folder, чи Archive.

Результат: тестування стає простішим і більш ізольованим.

6. Гнучка робота з різними типами об’єктів

Шаблон “Visitor” дозволяє працювати з різнотипними елементами (File, Folder, Archive) в одній структурі. Через клас ObjectStructure можна виконати операції над усіма елементами, незалежно від їх типу.

Результат: забезпечується гнучкість у роботі з колекціями елементів різних типів.

7. Спрощення підтримки

Логіка обробки винесена в окремі класи-відвідувачі, що спрощує її модифікацію. Для внесення змін достатньо відредагувати відповідний відвідувач.

Результат: проект легше підтримувати та модифікувати.

Висновок: отже, у ході виконання лабораторної роботи було реалізовано шаблон проектування Visitor, який полегшує управління складними операціями в проекті, дозволяючи централізувати логіку та забезпечуючи простоту розширення. Клієнтський код залишається чистим і легким для читання, оскільки всі операції виконуються через Visitor, приховуючи внутрішні деталі реалізації.

Посилання на код: https://github.com/SofiiaKashubiak/Archivator-project.git