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 №8

«ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»»

Варіант 23

Виконала студентка групи ІА–13:

Сиваченко Дар'я Євгенівна

Перевірив :

Мягкий Михайло Юрійович

Київ 2023

Тема: Project Management software

Хід роботи

Шаблон "Composite" може бути досить ефективним для реалізації, оскільки він дозволяє об'єднати об'єкти в деревоподібні структури для роботи з ними як зі складовими частинами. застосувати шаблон "Composite" для групування користувачів, завдань і ресурсів у відповідні групи, дозволяючи виконувати операції на цій структурі як на одному цілому.

Основні складові шаблону "Composite":

Component Interface: Це інтерфейс, який визначає базовий метод showDetails(), який реалізують всі класи, які можуть бути частиною композиції.

Leaf Components (User, Task, Resource, Comment): Класи, які представляють кінцеві об'єкти, які не можуть містити інші об'єкти в собі. Вони реалізують інтерфейс Component та надають реалізацію методу showDetails(), яка повертає деталі конкретного об'єкта.

Composite Component (Group): Цей клас також реалізує інтерфейс Component, але він містить колекцію інших об'єктів типу Component. Це дозволяє додавати і видаляти об'єкти будь-якого типу (User, Task, Resource, Comment) як частини цієї структури. Метод showDetails() у композиті додає деталі про саму групу та про всі її компоненти.

Main Class (Main): У головному класі створено об'єкти різних класів (користувачів, завдання, ресурси, коментарі) і групуємо їх за допомогою об'єктів класу Group, які представляють собою композицію. Потім ми додаємо ці групи до іншої головної групи, створюючи ієрархічну структуру.

Шаблон "Composite" дозволяє об'єднати об'єкти різних класів у структуру дерева, що дозволяє працювати з ними уніфіковано. Коли потрібно отримати деталі будь-якого рівня композиції,можна викликати метод showDetails() для головного об'єкта Main Group, який рекурсивно викликає цей метод для всіх його компонентів, виводячи деталі кожного об'єкта, включаючи всі підкомпоненти у вигляді структурованої інформації.Дана реалізація представлена разом зі звітом.

Висновок:

Лабораторна робота показала використання шаблону "Composite" для організації ієрархічної структури об'єктів. У контексті наданих класів, шаблон "Composite" був застосований для об'єднання об'єктів різних класів (користувачів, завдань, ресурсів, коментарів) у групи, створюючи таким чином єдину структуру даних.Цей підхід дозволяє працювати з різнорідними об'єктами як зі складовими частинами, використовуючи загальний інтерфейс. Класи-листки (leaf components) представляють кінцеві об'єкти, в той час як композитні класи групують ці об'єкти у ієрархічну структуру, що дозволяє виконувати операції з усією структурою як з єдиною сутністю.Застосування шаблону "Composite" робить код більш гнучким і розширюваним, дозволяючи працювати зі складними структурами об'єктів у єдиному стилі програмування. Цей підхід забезпечує однорідний доступ до даних та оперування з об'єктами різного типу через загальний інтерфейс, що полегшує управління і обробку цих об'єктів у великих програмних системах.

Відповіді на контрольні запитання

1.Навіщо використовується шаблон «композитний об'єкт» ?

Шаблон "композитний об'єкт" використовується для створення ієрархічних структур, що дозволяють об'єднувати об'єкти в складові частини більших об'єктів. Основна мета цього шаблону - дозволити операції, які працюють з одиночними об'єктами, працювати зі складними структурами таким самим чином.

2**. Який виграш від використання шаблону «Flyweight»?**

Основні вигоди від використання шаблону "Flyweight":

- Економія пам'яті: Цей шаблон дозволяє ефективно використовувати пам'ять, особливо у випадках, коли у вас велика кількість однакових або подібних об'єктів. Він використовує пул об'єктів, де спільні характеристики зберігаються один раз, а не для кожного окремого екземпляра, що дозволяє значно зменшити обсяг пам'яті, необхідний для програми.
- Підвищення продуктивності: Оптимізація використання пам'яті також може позитивно позначитися на продуктивності програми. Зменшення обсягу пам'яті, який використовується, може призвести до зменшення використання ресурсів системи, що може покращити час відповіді програми.

- Реюзабельність об'єктів: Шаблон "Flyweight" сприяє використанню одного екземпляра об'єкта для різних контекстів, що дозволяє підвищити реюзабельність і покращити швидкодію програми.
- Спрощення структури даних: Використання "Flyweight" може спростити структуру даних програми, оскільки він розділяє дані на внутрішні та зовнішні частини, що дозволяє зберігати загальні дані централізовано і використовувати їх у багатьох об'єктах.

3. У чому основне призначення використання шаблону «Інтерпретатор» ?

Шаблон "Інтерпретатор" (Interpreter) використовується для вирішення задач інтерпретації та виконання виразів або мовних конструкцій. Основне призначення цього шаблону полягає в створенні механізму обробки мови або специфічного формату даних.

4. Які переваги шаблону «відвідувач»?

Переваги використання шаблону "відвідувач":

- **Розділення алгоритму та структури:** Шаблон "відвідувач" дозволяє розділити алгоритми від структури об'єктів, що робить код більш модульним та зрозумілим. Алгоритми представлені класами "відвідувачів", які відвідують різні об'єкти, не залежать від їх конкретних класів.
- **Легка додаткова функціональність:** Додавання нових операцій для обробки об'єктів стає простішим без зміни самого класу об'єктів. Можливість створювати нові класи "відвідувачів" дозволяє легко додавати нові функціональні можливості.
- **Підтримка відкритості / закритості принципу проектування:** Шаблон "відвідувач" сприяє принципу відкритості / закритості, де класи об'єктів залишаються закритими для змін, але відкритими для розширення.
- **Легка підтримка нових типів об'єктів**: Шаблон "відвідувач" дозволяє легко додавати нові типи об'єктів, адже для кожного нового типу потрібно створити тільки новий клас "відвідувача", який реалізує потрібну функціональність для цього типу.
- **Зменшення дублювання коду:** Код алгоритмів обробки об'єктів зосереджений у відповідних класах "відвідувачів". Це дозволяє уникнути дублювання коду для обробки однакових операцій над різними типами об'єктів.

5. Які недоліки шаблону «відвідувач»?

Недоліки:

1. **Збільшення складності коду**: Додаткові класи "відвідувачів" і інтерфейси для об'єктів можуть призвести до збільшення кількості класів у програмі, що може зробити код складнішим для розуміння та підтримки.
2. **Залежність від структури об'єктів:** Шаблон "відвідувач" вимагає, щоб об'єкти визначали методи прийому "відвідувачів". Це може призвести до прив'язаності класів до конкретних відвідувачів, порушуючи принцип відкритості / закритості.
3. **Ускладнює додавання нових класів об'єктів**: Якщо потрібно додати нові класи об'єктів, то відповідно потрібно буде створити нові класи "відвідувачів" для обробки цих об'єктів. Це може вимагати значних змін у вже наявному коді.
4. **Порушення інкапсуляції**: Деякі інтерфейси "відвідувачів" можуть потребувати доступу до внутрішніх деталей об'єктів, що може порушити принцип інкапсуляції.
5. **Обмеженість в динамічних змінах поведінки**: Шаблон "відвідувач" визначає поведінку об'єктів під час компіляції, що ускладнює динамічні зміни поведінки на етапі виконання.