ЛАБОРАТОРНА РОБОТА №6. ШАБЛОНИ ПОВЕДІНКИ. ШАБЛОНИ STRATEGY, CHAIN OF RESPONSIBILITY, VISITOR

Мета: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Strategy, Chain of Responsibility та Visitor.

Довідка

Strategy

Проблема. Спроектувати змінювані, але надійні алгоритми або стратегії.

Рішення. Визначити для кожного алгоритму або стратегії окремий клас зі стандартним інтерфейсом "Strategy".

Створюється декілька класів "ConcreteStrategy", кожен з яких містить один і той же поліморфний метод "AlgorithmInterface". Об'єкт стратегії зв'язується з контекстним об'єктом (тим об'єктом, до якого застосовується алгоритм).

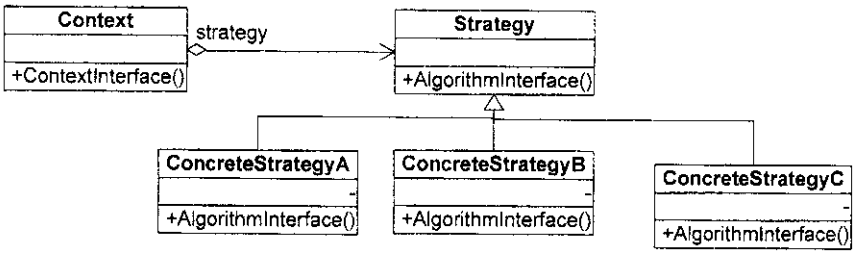

Рис.1. Структура шаблону Strategy

Chain of Responsibility

Проблема. Запит повинен бути оброблений декількома об'єктами.

Рішення. Зв'язати об'єкти – одержувачі запиту в ланцюжок і передавати запит вздовж цього ланцюжка, поки він не буде оброблений. "Handler" визначає інтерфейс для обробки запитів, і, можливо, реалізує зв'язок з наступником. "ConcreteHandler" обробляє запит, за який відповідає. Маючи доступ до свого наступника "ConcreteHandler" направляє запит до нього, якщо не може обробити запит сам.

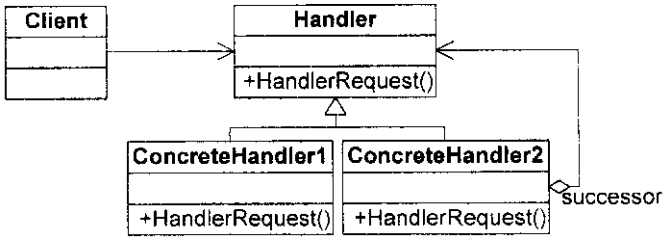

Рис.2. Структура шаблону Chain of Responsibility

Логічним є застосування цього шаблону, якщо є більше одного об'єкта, здатного обробити запит і обробник заздалегідь невідомий (і повинен бути знайдений автоматично) або якщо весь набір об'єктів, які здатні обробити

запит, повинен задаватися автоматично.

Шаблон послаблює зв'язаність (об'єкт не зобов'язаний "знати", хто саме опрацює його запит). Але немає гарантій, що запит буде оброблений, оскільки він не має явного одержувача.

Visitor

Проблема. Над кожним об'єктом деякої структури виконується операція. Визначити нову операцію, не змінюючи класи об'єктів.

Рішення. Клієнт, який використовує даний шаблон, повинен створити об'єкт класу "ConcreteVisitor", а потім відвідати кожен елемент структури. "Visitor" оголошує операцію, "VisitConcreteElement" для кожного класу "ConcreteElement" (ім'я та сигнатура даної операції ідентифікують клас, елемент якого відвідує "Visitor" – тобто, відвідувач може звертатися до елементу безпосередньо). "ConcreteVisitor" реалізує всі операції, оголошення в класі "Visitor". Кожна операція реалізує фрагмент алгоритму, визначеного для класу відповідного об'єкта в структурі. Клас "ConcreteVisitor" надає контекст для цього алгоритму і зберігає його локальний стан. "Element" визначає операцію "Accept", яка приймає "Visitor" як аргумент, "ConcreteElement" реалізує операцію "Accept", яка приймає "Visitor" як аргумент. "ObjectStructure" може перерахувати свої аргументи і надати відвідувачеві високорівнева інтерфейс для відвідування своїх елементів.

Логічним є використання цього шаблону, якщо в структурі присутні об'єкти багатьох класів з різними інтерфейсами, і необхідно виконати над ними операції, що залежать від конкретних класів, або якщо класи, що визначають структуру об'єктів змінюються рідко, але нові операції над цією структурою додаються часто.

Шаблон спрощує додавання нових операцій, об'єднує споріднені операції в класі "Visitor". При цьому ускладнюється додавання нових класів "ConcreteElement", оскільки потрібно оголошення нової абстрактної операції в класі "Visitor".

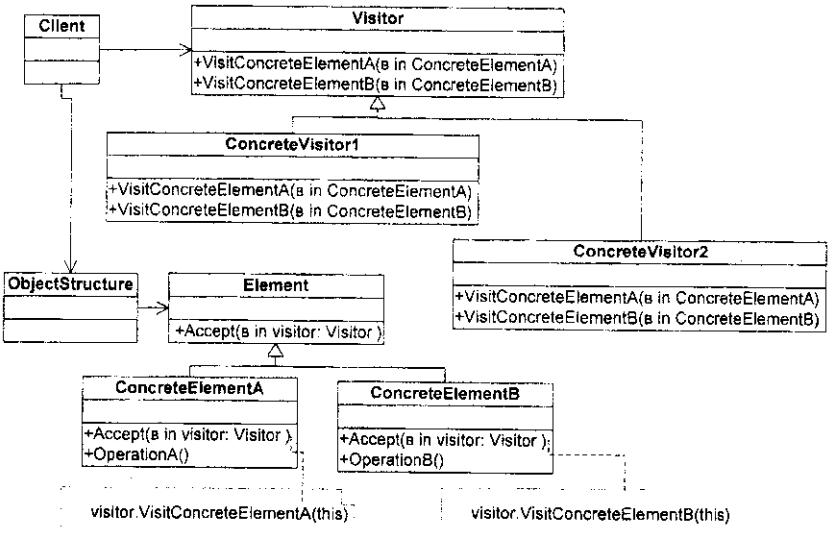

Рис.3. Структура шаблону Visitor

Завдання

1. Повторити шаблони поведінки для проектування ПЗ. Знати загальну характеристику шаблонів поведінки та призначення кожного з них.
2. Детально вивчити шаблони поведінки для проектування ПЗ — Strategy, Chain of Responsibility та Visitor. Для кожного з них:
   - вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
   - знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
   - вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
   - вміти розпізнавати Шаблон в UML діаграмі класів та будувати сирцеві коди Java-класів, що реалізують шаблон.
3. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork6. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.2). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес-логіку закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи.
4. За допомогою автоматизованих засобів виконати повне документування розроблених класів (також методів і полів), при цьому документація має в достатній мірі висвітлювати роль певного класу в загальній структурі Шаблону та особливості конкретної реалізації.

Варіанти завдання

Номер варіанту завдання обчислюється як залишок від ділення номеру залікової книжки на 10.

0. Визначити специфікації класу, який містить масив цілих чисел та метод його сортування. Забезпечити можливість динамічної зміни алгоритму та напрямку сортування шляхом зовнішньої параметризації.
1. Визначити специфікації класу, який містить таблицю та метод її відображення у вигляді діаграми. Забезпечити можливість динамічної зміни типу діаграми шляхом зовнішньої параметризації.
2. Визначити специфікації класу, який містить математичну функцію та метод її відображення у вигляді графіка. Забезпечити можливість динамічної зміни системи координат графіка (декартова, полярна тощо) шляхом зовнішньої параметризації.
3. Визначити специфікації класів, що реалізують контейнери для цілих чисел та текстових строк з можливістю їх сортування. Забезпечити можливість динамічної зміни алгоритму сортування шляхом зовнішньої параметризації. Реалізація алгоритму сортування має бути незалежною від типу даних, що сортуються.

4. Визначити специфікації класів, що реалізують елементи графічного інтерфейсу користувача — панелі (композит) та кнопки (компонент). Реалізувати децентралізований механізм обробки події переміщення курсору миші. Кількість компонентів інтерфейсу, які реагують на цю подію, може змінюватись динамічно.
5. Визначити специфікації класів, що реалізують обробку HTTP-запитів різних типів (наприклад GET та POST). Реалізувати можливість динамічної зміни кількості обробників. Забезпечити децентралізацію та слабку зв'язаність обробників.
6. Визначити специфікації класів для елементу ігрового поля (комірки) та самого простору. Забезпечити слабку зв'язаність елементів. Реалізувати децентралізований механізм сумісної зміни стану елементів.
7. Визначити специфікації класів, що реалізують елементи графічного інтерфейсу користувача — панелі (композит) та кнопки (компонент). Реалізувати механізм додаткових операцій над структурою графічного інтерфейсу без зміни/її елементів. В якості ілюстрації такого механізму розробити операцію підрахунку кількості елементів одного типу.
8. Визначити специфікації класів, що реалізують елементи структури комп'ютера (процесор, пам'ять, відеокарта тощо). Реалізувати механізм додаткових операцій над структурою комп'ютера без зміни її елементів. В якості ілюстрації такого механізму розробити операцію визначення потужності, що потребляє комп'ютер.
9. Визначити специфікації класів, що реалізують елементи мережевої структури (кабель, сервер, робоча станція). Реалізувати механізм додаткових операцій над мережевою структурою без зміни її елементів. В якості ілюстрації такого механізму розробити операцію визначення кошторису такої структури.

Протокол

Протокол має містити титульну сторінку (з номером залікової книжки), завдання, роздруківку діаграми класів, розроблений програмний код та згенеровану документацію в форматі JavaDoc.