ЛАБОРАТОРНА РОБОТА №7. ШАБЛОНИ ПОВЕДІНКИ. ШАБЛОНИ MEMENTO, STATE, COMMAND, INTERPRETER

Мета: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Memento, State, Command та Interpreter.

Довідка

Memento

Проблема. Необхідно зафіксувати стан об'єкту для реалізації, наприклад, механізму відкату.

Рішення. Зафіксувати і винести (не порушуючи інкапсуляції) за межі об'єкта його внутрішній стан так, щоб згодом можна було його відновити в об'єкті. "Memento" зберігає внутрішній стан об'єкта "Originator" і забороняє доступ до себе всім іншим об'єктам окрім "Originator", який має доступ до всіх даних для відновлення в колишньому стані. "Caretaker" може лише передавати "Memento" іншим об'єктам. "Originator" створює "Memento", що містить знімок поточного внутрішнього стану і використовує "Memento" для відновлення внутрішнього стану. "Caretaker" відповідає за збереження "Memento", при цьому не робить ніяких операцій над "Memento" і не досліджує його внутрішній вміст. "Caretaker" запитує "Memento" у "Originator", деякий час тримає його у себе, а потім повертає "Originator".

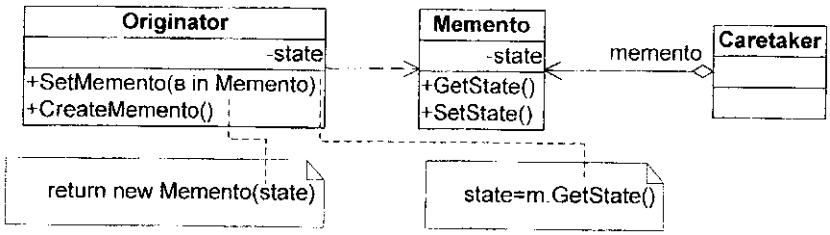

Рис.1. Структура шаблону Memento

При використанні шаблону не розкривається інформація, яка доступна тільки "Originator", спрощується структура "Originator". Але з використанням "Memento" можуть бути пов'язані значні витрати, якщо "Originator" повинен копіювати великий обсяг інформації, або якщо копіювання повинно проводитися часто.

State

Проблема. Варіювати поведінку об'єкта в залежності від його внутрішнього стану.

Рішення. Клас "Context" делегує запити, які залежать від стану, поточному об'єкту "ConcreteState" (зберігає екземпляр підкласу "ConcreteState", яким визначається поточний стан), і визначає інтерфейс, що представляє інтерес для клієнтів. "ConcreteState" реалізує поведінку, асоційовану з якимось станом об'єкта "Context". "State" визначає інтерфейс для інкапсуляції поведінки,

асоційованої з конкретним екземпляром "Context".

Шаблон локалізує залежну від стану поведінку і ділить її на частини, що відповідають станам, переходи між станами стають явними.

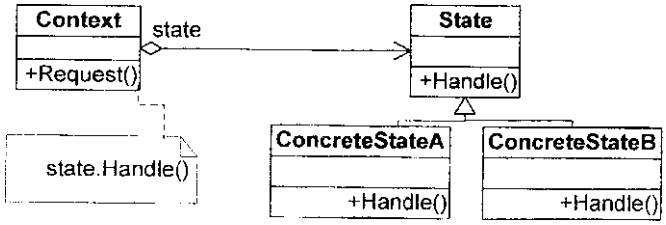

Рис.2. Структура шаблону State

Command

Проблема. Необхідно надіслати об'єкту запит, не знаючи про те, виконання якої операції запитане і хто буде одержувачем.

Рішення. Інкапсулювати запит як об'єкт. "Client" створює об'єкт "ConcreteCommand", який викликає операції одержувача для виконання запиту, "Invoker" відправляє запит, виконуючи операцію "Command" Execute(). "Command" оголошує інтерфейс для виконання операції, "ConcreteCommand" визначає зв'язок між об'єктом "Receiver" і операцією Action(), і, крім того, реалізує операцію Execute() шляхом виклику відповідних операцій об'єкта "Receiver". "Client" створює екземпляр класу "ConcreteCommand" і встановлює його одержувача, "Invoker" звертається до команди для виконання запиту, "Receiver" (будь-який клас) має інформацію про способи виконання операцій, необхідних для виконання запиту.

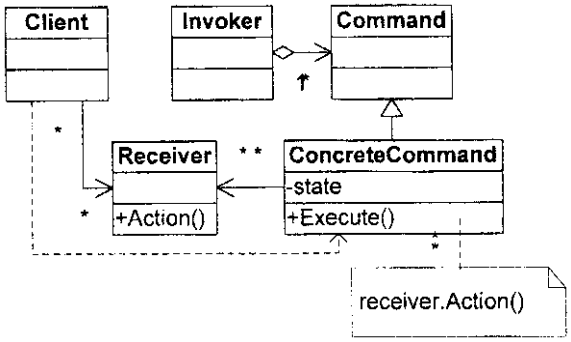

Рис.3. Структура шаблону Command

Шаблон Command розриває зв'язок між об'єктом, який ініціює операції, і об'єктом, що має інформацію про те, як її виконати, крім того створюється об'єкт "Command", який можна розширювати і маніпулювати ним як об'єктом.

Interpreter

Проблема. Існує підвладна змінам задача, що зустрічається часто.

Рішення. Створити інтерпретатор, який вирішує цю задачу.

Завдання пошуку рядків за зразком може бути вирішено за допомогою створення інтерпретатора, що визначає граматику мови. "Client" будує речення у вигляді абстрактного синтаксичного дерева, у вузлах якої знаходяться об'єкти

класів "TerminalExpression" і "NontemiinalExpression" (рекурсивне), потім "Client" ініціалізує контекст і викликає операцію Interpret(Context). На кожному вузлі типу "TerminalExpression" реалізується операція Interpret(Context) для кожного підвиразу. Для класу "NonterminalExpression" операція Interpret(Context) визначає базу рекурсії. "AbstractExpression" визначає абстрактну операцію Interpret(Context), загальну для всіх вузлів у абстрактному синтаксичному дереві. "Context" містить інформацію, глобальну по відношенню до інтерпретатору.

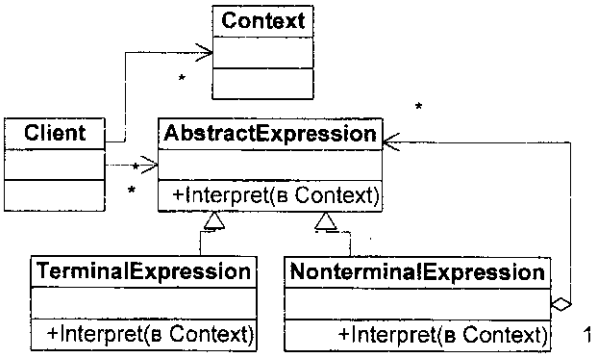

Рис.4. Структура шаблону Interpreter

Завдяки шаблону граматику стає легко розширювати і змінювати, реалізації класів, що описують вузли абстрактного синтаксичного дерева схожі (легко кодуються). Можна легко змінювати спосіб обчислення виразів. Але важко супроводжувати граматику з великим числом правил.

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 – Memento, State, Command та Interpreter. 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.labwork7. 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 8.

0. Визначити специфікації класів, які подають у векторному редакторі графічні елементи (коло, трикутник тощо) з різними атрибутами (колір, позиція, розмір, тощо). Реалізувати механізм збереження/встановлення стану елемента.
1. Визначити специфікації класу, що подає персонажа в ігровому просторі з необхідними атрибутами (позиція персонажу, склад артефактів, рівень "здоров'я" тощо). Реалізувати механізм збереження/встановлення стану персонажа.
2. Визначити специфікації класів, які подають операції над таблицю в БД. Реалізувати механізм організації транзакцій при виконанні операцій над таблицею.
3. Визначити специфікації класу, що подає мережеве з'єднання протоколу TCP. Реалізувати зміну поведінки в залежності від стану з'єднання (LISTENING, ESTABLISHED, CLOSED) без використання громіздких умовних операторів.
4. Визначити специфікації класів, що подають інструменти малювання та робочий простір в графічному редакторі. Реалізувати механізм зміни реакції на натискання кнопки миші в залежності від вибраного інструменту. Уникати використання громіздких умовних конструкцій.
5. Визначити специфікації класів, що подають чергу HTTP-запитів на обробку. Реалізувати можливість виключення запитів з черги без обробки, та зміни позиції запиту через зміну значення пріоритету.
6. Визначити специфікації класів, що подають реакції на натискання пунктів меню та кнопок інструментальної панелі. Забезпечити можливість динамічної зміни реакції, а також формування макрореакцій (послідовність з наперед заданих реакцій).
7. Визначити специфікації класів для розбору алгебраїчних виразів з операціями +, *, /.

Протокол

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