УДК 004.4

А.В. Кушнір
Р.В. Маслій

АНАЛІЗ ПАТЕРНІВ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація
Використання патернів проектування це простий та зручний спосіб вирішити проблеми, які уже були вирішенні, при створенні програмного продукту.
Ключові слова: патерн проектування, об'єктно-орієнтоване програмування, об'єкт, клас, підклас.

Abstract
Using design patterns is a simple and convenient way to solve problems that have already been solved when creating a software product.
Keywords: pattern design, object-oriented programming, object, class, subclass.

Вступ

Створюючи об'єкт, у тому числі й програмний продукт, розробник часто стикається із завданнями, які вже хто-небудь вирішив. У 70-ті роки XX ст. архітектор Кристофер Александр запропонував шаблони проектування будинків та міст. Через десятиліття ця ідея переросла у процесі розроблення програмного забезпечення до шаблонів проектування інтерфейсу користувача, запропонованих Вардом Каннінгемом та Кентом Беком. Далі ідея була активно підхоплена та розвинута у вигляді каталогу патернів ООП. Цей каталог став дуже популярним серед розробників. Ідея використання не тільки коду, а й архітектурних та проектних рішень виявилася настільки успішною, що сьогодні патерни проектування широко застосовуються в різних методиках розроблення програмного забезпечення [1].

Аналіз патернів проектування

У роботі під патернами проектування об'єктно-орієнтованих систем розуміють опис взаємодії об'єктів і класів, адаптованих для вирішення спільної задачі проектування в конкретному контексті.

Існує багато патернів розроблення програмних систем, які відмінні сферою застосування, масштабом, змістом, стилем опису [1].

Найбільш поширені патерни групуються у три категорії: ітератор поведінкові та структурні.

Породжуючі – патерни, що надають рекомендації та техніки для створення нових об'єктів [2]. Існують такі породжуючи патерни:

- абстрактна фабрика (Abstract Factory) – надає інтерфейс для створення груп зв'язаних або залежних об'єктів, не вказуючи на їх конкретний клас;
- будівник (Builder) – розділяє створення складного об'єкта та ініціалізацію його стану так, що схожий процес побудови може створити об'єкти з різними станами;
- фабричний метод (Factory Method) – визначає інтерфейс для створення об'єкта, але дозволяє підкласам вирішувати, який клас ініціалізувати. Дозволяє делегувати створення об'єкта підкласами;
- прототип (Prototype) – визначає декілька видів об'єктів, щоб при створенні використовувати об'єкт-прототип і створює нові об'єкти, копіюючи прототип;
- одинак (Singleton) – гарантує, що клас має тільки один екземпляр і надає глобальну точку доступу до нього [3].

Поведінкові – патерни, що надають рекомендації для реалізації тої чи іншої поведінкової функції існуючого об'єкта [2]. Є такі поведінкові патерни:

- знімок (Momento) – не порушуючи інкапсуляцію, визначає і зберігає внутрішній стан об'єкта і дозволяє пізніше відновити об'єкт в цьому стані;

- ланцюг обов’язків (Chain of responsibility) – уникає зв’язування відправника запиту з його отримувачем, надаючи можливість опрацювати запит більше ніж одному об’єкту. Зв’язує об’єкти-отримувачі і передає запит по ланцюгу до тих пір, поки об’єкт не опрацює його;

- спостерігач (Observer) – визначає залежність «один до багатьох» між об’єктами так, що коли один об’єкт змінює свій стан, всі залежні об’єкти сповіщуються і оновлюються автоматично;

- команда (Command) – інкапсулює запит у вигляді об’єкта, дозволяючи передавати їх клієнтам в якості параметрів;

- стан (State) – дозволяє об’єкту змінювати свою поведінку в залежності від внутрішнього стану;

- інтерпретатор (Interpreter) – отримуючи формальну мову, визначає представлення його граматики та інтерпретатор, що використовує це представлення для обробки вираження мови;

- стратегія (Strategy) – визначає групу алгоритмів, інкапсулює їх та робить взаємозамінними. Дозволяє змінювати алгоритм незалежно від клієнтів, що його використовують;

- ітератор (Iterator) – надає спосіб послідовного доступу до елементів множини, незалежно від його внутрішнього устрою;

- шаблонний метод (Template method) – визначає алгоритм, деякі етапи якого делегуються підкласами. Дозволяє підкласам перевизначити ці етапи, не змінюючи структури алгоритму;

- посередник (Mediator) – визначає об’єкт, що інкапсулює спосіб взаємодії об’єктів. Забезпечує слабкий зв’язок, позбавляючи об’єкти від необхідності прямо посилатись один на одного і надає можливість незалежно змінювати їх взаємодію;

- відвідувач (Visitor) – представляє собою операцію, яка буде виконана над об’єктами групи класів. Дає можливість визначити нову операцію без зміни кода класів, над якими ця операція здійснюється [3].

Структурні – даний тип патернів розглядає питання взаємодії між собою існуючих об’єктів [2]:

- адаптер (Adapter) – конвертує інтерфейс класу в інтерфейс, який очікує клієнт. Дозволяє класам з різними інтерфейсами працювати разом;

- замісник (Proxy) – надає заміну об’єкта для контролю доступу до нього;

- міст (Bridge) – розділяє абстракцію та реалізацію так, щоб вони могли змінюватись незалежно;

- компонувальник (Composite) – компонує б’єкти в деревовидну структуру, представляючи їх у вигляді ієрархії. Дозволяє клієнту однаково звертатись як до окремого об’єкту, так і до цілого піддерева;

- декоратор (Decorator) – динамічно надає об’єкту додаткові можливості. Представляє собою гнучку альтернативу наслідування для розширення функціоналу;

- легковаговик (Flyweight) – завдяки спільному використанню, підтримує ефективну роботу з великою кількістю об’єктів [3].

Висновки

Сьогодні основна роль патернів проектування – повторне використання досвіду в різних областях розроблення програмного забезпечення; усунення комунікаційного бар’єру всередині команди розробників і між ними; підвищення якості створюваного продукту за рахунок використання перевірених роками рішень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1.Алексенко Ольга Василівна «Технології програмування та створення програмних продуктів» : конспект лекцій для студ. напряму підготовки 6.050101 "Комп'ютерні науки" усіх форм навчання / О. В. Алексенко. — Суми : СумДУ, 2013. — 133 с.

2. Подоба В. Патерни Програмування: Що таке патерни та їхні типи? [Електронний ресурс] / Віталій Подоба. – 2014. – Режим доступу до ресурсу: http://www.vitaliypodoba.com/2014/06/programming-patterns-intro/.

3. Шпаргалка по шаблонам проектирования [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://habr.com/ru/post/210288/.

Кушнір Анастасія Володимирівна – студентка групи 2СІ-15б, факультет комп’ютерних систем і автоматики, Вінницький національний технічний університет, Вінниця, e-mail: fkca.2ci15.kav@gmail.com

Науковий керівник: ***Маслій Роман Васильович —*** к. т. н, доцент кафедри АІІТ, факультет комп’ютерних систем та автоматики, Вінницький національний технічний університет, м.Вінниця, e-mail: romas4580@gmail.com.

Kushnir Anastasiia Volodymyrivna – Faculty of computer systems and automation, Vinnytsia national technical University, Vinnytsia, email: fkca.2ci15.kav@gmail.com

Supervisor: **Maslii Roman V.** – Phd, Associate Professor, Department of Computer Systems and Automation, Vinnytsia National Technical University. Vinnitsa, e-mail: romas4580@gmail.com.