МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ  
«ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ЛУЦЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ»

Випускова циклова комісія Компʼютерних систем та інформаційних технологій

**Звіт**

**з практичної роботи №7**

з дисципліни “Технології розробки програмного забезпечення”

на тему: “Вступ до патернів проектування”

Виконав: студент **2** курсу, групи **22-ІСТ**

**Синиця Назар**

Викладач: Володимир Остапюк

Термін здачі: 27-11-2024

Здано:

Луцьк – 2024

**Завдання**

**Дослідити основи патернів проектування та дати розгорнуті відповіді на запитання:**

1. Що таке патерн?
2. Історія патернів.
3. Навіщо знати патерни?
4. Критика патернів.
5. Класифікація патернів.

**Висновок**

**1.Що таке патерн?**

**Патерн проектування** — це типовий спосіб вирішення певної проблеми, що часто зустрічається при проектуванні архітектури програм.

На відміну від готових функцій чи бібліотек, патерн не можна просто взяти й скопіювати в програму. Патерн являє собою не якийсь конкретний код, а загальний принцип вирішення певної проблеми, який майже завжди треба підлаштовувати для потреб тієї чи іншої програми.

Патерни часто плутають з алгоритмами, адже обидва поняття описують типові рішення відомих проблем. Але якщо алгоритм — це чіткий набір дій, то патерн — це високорівневий опис рішення, реалізація якого може відрізнятися у двох різних програмах.

Якщо провести аналогії, то алгоритм — це кулінарний рецепт з чіткими кроками, а патерн — інженерне креслення, на якому намальовано рішення без конкретних кроків його отримання.

**2.Історія патернів.**

Хто вигадав патерни? Це хороше, але не зовсім коректне запитання, бо патерни не вигадують, а радше «відкривають». Це не якісь супер-оригінальні рішення, а, навпаки, типові способи вирішення однієї і тієї ж проблеми, що часто повторюються з невеликими варіаціями.

Концепцію патернів вперше описав Крістофер Александер у книзі «[**Мова шаблонів. Міста. Будівлі. Будівництво**](https://refactoring.guru/uk/pattern-language-book)». У книзі описано «мову» для проектування навколишнього середовища, одиниці якого — шаблони (або патерни, що ближче до оригінального терміна patterns) — відповідають на архітектурні запитання: якої висоти потрібно зробити вікна, скільки поверхів має бути в будівлі, яку площу в мікрорайоні відвести для дерев та газонів.

Ідея видалася привабливою четвірці авторів: Еріху Гаммі, Річарду Хелму, Ральфу Джонсону, Джону Вліссідесу. У 1994 році вони написали книгу «**[Патерни проектування: повторно використовувані елементи архітектури об’єктно-орієнтованого програмного забезпечення](https://refactoring.guru/uk/gof-book)**», до якої увійшли 23 патерни, що вирішують різні проблеми об’єктно-орієнтованого дизайну. Назва книги була занадто довгою, щоб хтось зміг її запам’ятати. Тому незабаром усі стали називати її «book by the gang of four», тобто «книга від банди чотирьох», а потім і зовсім «GoF book».

З того часу було знайдено десятки інших об’єктних патернів. «Патерновий» підхід став популярним і в інших галузях програмування, тому зараз можна зустріти різноманітні патерни також за межами об’єктного проектування.

**3.Навіщо знати патерни?**

Ви можете цілком успішно працювати, не знаючи жодного патерна. Більше того, ви могли вже не раз реалізувати який-небудь з патернів, навіть не підозрюючи про це.

Але якраз свідоме володіння інструментом відрізняє професіонала від аматора. Ви можете забити цвях молотком, а можете й дрилем, якщо дуже сильно постараєтесь. Але професіонал знає, що головна фішка дриля зовсім не в цьому. Отже, навіщо ж знати патерни?

* **Перевірені рішення.** Ви витрачаєте менше часу, використовуючи готові рішення, замість повторного винаходу велосипеда. До деяких рішень ви могли б дійти й самотужки, але багато які з них стануть для вас відкриттям.
* **Стандартизація коду.** Ви робите менше прорахунків при проектуванні, використовуючи типові уніфіковані рішення, оскільки всі приховані в них проблеми вже давно знайдено.
* **Загальний словник програмістів.** Ви вимовляєте назву патерна, замість того, щоб годину пояснювати іншим програмістам, який крутий дизайн ви придумали і які класи для цього потрібні.

**4.Критика патернів.**

Патерни були описані більше 20-ти років тому, тому тільки ледачий не встиг кинути в них камінь. Давайте розглянемо найпопулярнішу критику.

**Розпорки для слабкої мови програмування**

Потреба в патернах з’являється тоді, коли люди вибирають для свого проекту мову програмування з недостатнім рівнем абстракції. В цьому випадку, патерни — це розпорки, які надають цій мові суперздібності.

Наприклад, патерн [**Стратегія**](https://refactoring.guru/uk/design-patterns/strategy) в сучасних мовах можна реалізувати простою анонімною (лямбда) функцією.

**Неефективні рішення**

Патерни намагаються стандартизувати підходи, які й так вже широко використовуються. Така стандартизація здається деяким людям догмою і вони починають усюди реалізовувати патерни «як в книжці», не пристосовуючи їх до реалій проекту.

**Невиправдане застосування**

Якщо у тебе в руках молоток, то всі предмети навколо починають нагадувати цвяхи.

Схожа проблема виникає у новачків, які тільки-но познайомилися з патернами. Вникнувши в патерни, людина намагається застосувати свої знання всюди. Навіть там, де можна було б обійтися більш простим кодом.

**5.Класифікація патернів.**

Патерни відрізняються за рівнем складності, деталізації та охоплення проектованої системи. Проводячи аналогію з будівництвом, ви можете підвищити безпеку на перехресті, встановивши світлофор, а можете замінити перехрестя цілою автомобільною розв’язкою з підземними переходами.

Найбільш низькорівневі та прості патерни — ідіоми. Вони не дуже універсальні, позаяк мають сенс лише в рамках однієї мови програмування.

Найбільш універсальні — архітектурні патерни, які можна реалізувати практично будь-якою мовою. Вони потрібні для проектування всієї програми, а не окремих її елементів.

Крім цього, патерни відрізняються і за призначенням. У цій книзі буде розглянуто три основні групи патернів:

* **Породжуючі патерни** піклуються про гнучке створення об’єктів без внесення в програму зайвих залежностей.
* **Структурні патерни** показують різні способи побудови зв’язків між об’єктами.
* **Поведінкові патерни** піклуються про ефективну комунікацію між об’єктами.

**Висновок**

Під час виконання роботи я зрозумів що патерни є важливим інструментом для вирішення типових проблем, що виникають у різних галузях, зокрема в програмуванні, дизайні та архітектурі. Вони допомагають створювати ефективні, структуровані і зручні для обслуговування системи.