**МОДУЛЬ 1**

***Лабораторна робота 1.3***

**ДОСЛІДЖЕННЯ ВЗАЄМОЗВ'ЯЗКУ ШАБЛОНІВ ПРОЕКТУВАННЯ ТА ОСНОВНИХ ПРИНЦИПІВ ПРОЕКТУВАННЯ**

**Завдання**

1. Спроектувати, представити у вигляді UML діаграми(-ам) функціонал згідно з варіантом. На діаграмі виділити використаний шаблон проектування.
2. Розробити програму на мові C#, яка відповідає вимогам у варіанті. Для демонстрації роботи використати текстові повідомлення на консолі через операції вводу-виводу.
3. Діаграма(-и) та вихідний код повинні відповідати основним принципам проектування: OOP, SOLID, Law of Demeter (LoD), DRY, YAGNI, KISS, cohesion – coupling, inheritance with caution.
4. Для отримання балів, що відповідають «задовільно» необхідно реалізувати завдання, відповідно варіанту (табл. 1) з одним шаблоном проектування на власний вибір.
5. Для отримання балів, що відповідають «добре» необхідно у реалізацію завдання з табл. 1 додати імплементацію ще мінімум одного чи більше шаблонів проектування. Пояснити вибір, а також існуючі альтернативи. Які переваги дають шаблони в порівнянні з реалізацією такого ж завдання в попередній лабораторній роботі? Необхідною умовою є використання репозиторію Git. Початковий проект – попередня лабораторна робота. Показати та пояснити зроблені коміти (мінімум 3). Локальний репозиторій повинен бути обов’язково зв’язаний з віддаленим приватним репозиторієм.
6. Для отримання балів, що відповідають «відмінно» необхідно реалізувати реалізувати завдання у варіанті з табл. 1 з використанням 2-х і більше шаблонів проектування. Один із шаблонів повинен бути Observer, реалізований за допомогою механізму подій з виокремленням класів аргументів подій, генератора подій, спостерігачів та підписок (матеріал дисципліни ООП – делегати та події). Допускається в певних випадках використання інших шаблонів при достатній аргументації. Перед вибором шаблону варто проконсультуватися у викладача.

**Методичні рекомендації**

При виконанні роботи слід пам’ятати:

* вибір шаблону проектування повинен бути обгрунтованим. При цьому введення шаблону (-ів) не повинно порушувати відповідність коду принципам, вказаним у завданні в п.3. Якщо по-іншому неможливо ввести шаблон, варто або відрефакторити код так, щоб відновити відповідність принципам, або обрати інший шаблон;
* деякі шаблони проектування можуть мати кілька варіантів реалізації. Тому варто пояснити вибір саме цієї конкретної реалізації, та пояснити відмінності з іншими способами;
* при реалізації наблону Observer необхідно відокремити аргументи подій, генератор подій, спостерігачі та підписки, використавши механізм подій;
* різні характеристики/ сутності та специфічні операції не повинні бути в одному класі/ модулі;
* функціонал повинен бути легко розширюваним без необхідності внесення змін в існуючий код (чи при мінімальних змінах) – якщо в завданні не сказано протилежне;
* при необхідності реалізації однотипних операцій вварто використати абстракцію;
* код не повинен повторюватися і бути читабельним;
* коміти повинні бути атомарними та містити робочий код.

**Вимоги по реалізації застосування**

Використовувати багаторівневу архітектуру в даній лабораторній не є необхідним (але і не забороняється).

Головним критерієм якості є реалізація функціоналу з використанням шаблонів проектування та відповідність базовим принципам проектування. Використання в завданні шаблону проектування ні в якому разі не повинно призводити до порушення базові принципи проектування і створювати Code Smells. В цьому випадку варто провести рефакторинг коду. Якщо при цьому проблеми залишаються, варто ще раз задуматися про доцільність використання саме цього шаблону саме в цьому місці застосування.

Для сповіщення про зміни в стані об’єкту варто використовувати події, а також підписуватися чи відписуватися від них. Наприклад, у випадку зміни сума коштів на рахунку клієнт може отримувати смс-повідомлення; при цьому він може відмовитися від цих повідомлень. Чи коли хазяїну тварини бажано знати, коли його тварина зголодніла. Реалізація механізму подій в шаблоні Observer також ні в якому разі не повинна порушувати принципи проектування. Генерація події, підписки на події та інші компоненти цього шаблону повинні бути відокремлені. Також відокремленою повинна бути і підписка на події.

За необхідності можна додавати нові властивості, сутності та розширювати функціонал. Але ця необхідність повинна бути обґрунтована.

Управління та відповідь застосування здійснюється через консоль. Використання консолі, наприклад, для виведення повідомлень, потрібно винести за межі функціоналу предметної області. Тобто, вивід повідомлень на консоль повинен бути за межами методів, що виконують певні дії. Наприклад, вивід повідомлення про те, що тварина голодна повинна бути окремо від методу, який визначає стан тварини голодний. Методи класів предметної області мають змінювати певним чином стан об’єкту, а не просто виводити повідомлення на консоль (наприклад, на банківському рахунку змінюється сума коштів, чи тварина стає не голодна і т.д.).

Це ж саме стосується і використання шаблонів. Вони повинні бути використані саме для предметної області. Відповідно, в методах класів шаблону не повинно бути операцій вводу-виводу.

Сутності у завданнях (тварини, пристрої, ігри та ін.) створюються за схожими механізмами. Для цього можна використати деякі Creational шаблони;. Поведінка та умови цієї поведінки тварин, використання пристроїв, ігри теж мають деякі спільні риси. Тут можна використати як Behavioral, так і Structural шаблони.

Виконання операцій повинно здійснюватися тільки при настанні певних умов. Наприклад, багаторазовий виклик методу годування тварини не повинен приводити до її смерті від переїдання, або зняття коштів з рахунку не повинно призводити до негативного баланса. Для вирішення цієї задачі також можна використати шаблони проектування.

Для демонстрації роботи програми необхідно підготувати набір вхідних даних.

**Варіанти**

**Табл. 1**

|  |  |
| --- | --- |
| Номер  варіанту | Завдання |
| 6 | Розробити застосування, яке симулює поведінку тварин: собака, канарка, ящірка. Вони мають очі, лапи та крила і здатність їсти, бігати, ходити, співати, літати. При цьому тварина повинна їсти мінімум один раз на день, максимум – 5, через приблизно однакові проміжки часу. В інших випадках тварина помирає. Якщо у тварини попередній прийом їжі був більше 8 год тому, то вона не може бігати, співати чи літати, але може ходити, повзати. Тварини можуть жити у хазяїна або в зоомагазині. Незалежно від місця проживання за твариною передбачається догляд (прибирання). Якщо у тварини прибирається 1 раз на день чи частіше або вона живе на волі, вона є щасливою. Прибирання та щастя тварини не є необхідною умовою виживання тварини. Передбачити можливість випустити тварину на волі (в цьому разі за нею не доглядають). У хазяїна може бути кілька різних тварин. |

**Контрольні запитання**

1. Що таке Code Smells? Які ви знаєте види? Наведіть приклади. Чому це погано?
2. Що таке spaghetti code? Чому це вважається поганою практикою?
3. Які Code Smells належать до недоліків рівня застосування? Чому?
4. Які Code Smells належать до недоліків рівня класів? Чому?
5. Які Code Smells належать до недоліків рівня методів? Чому?
6. Для чого потрібні шаблони проектування? Які задачі/ проблеми вони допомагають вирішити?
7. Чи завжди потрібно використовувати багато шаблонів проектування? Чому?
8. Які існують групи шаблонів проектування? Чому вони розділяються саме так?
9. Що таке Creational шаблони? Для чого вони потрібні і як можуть бути використані?
10. Що таке Structural шаблони? Для чого вони потрібні і як можуть бути використані?
11. Що таке Behavioral шаблони? Для чого вони потрібні і як можуть бути використані?
12. Про що говорить GRASP?
13. Чим відрізняються централізовані репозиторії (Subversion) від розподілених (Git)?
14. Яка структура Git репозиторію?
15. Які види вказівників існують в Git? В чому різниця між ними?
16. Що таке коміт? Яка його структура? Як зробити коміт?
17. В яких статусах може перебувати файл репозиторія?
18. Які стани репозиторію існують і як вони пов’язані між собою?
19. Як можна переміщувати різні вказівники Git? Для яких цілей?
20. Як зв’язати локальний репозиторій з віддаленим і здійснювати синхронізацію?
21. Які архітектурні шаблони та стилі ви знаєте? Як вони розділені по категоріях?
22. В чому полягає поєднання архітектурних стилів?
23. Архітектура клієнт-сервер.
24. Компонентна архітектура.
25. Багаторівнева архітектура та N-рівнева архітектура.
26. Шина повідомлень.
27. Об’єктно-орієнтована архітектура.
28. Сервісно-орієнтована архітектура.
29. Що варто реалізовувати, як наскрізну функціональність? Чому?

**Примітка:** Також можуть бути задані питання про шаблони проектування з лабораторної роботи № 1-2.