Лабораторна робота №6-7
Вступ до паттернів
(максимум 10 балів)

Мета: Набуття практичних навичок роботи з деякими порівняно простими у реалізації паттернами в C# на прикладі модифікації вже розробленого консольного додатку, який імітує роботу того чи іншого підприємства або установи.

Завдання

1. Для лабораторної роботи 4-5 вдало реалізувати:
   - (1,5 бали) породжуючий паттерн проектування “Одинак”; його можна, наприклад, використати для ведення бази даних продажів чи переліку клієнтів тощо;
   - (3 бали) структурний паттерн проектування “Фасад”; його можна, наприклад, використати для формування переліку типових послуг чи послідовностей дій;
   - (3 бали) поведінковий паттерн проектування “Хранитель” (для зручності можна користуватись вкладенням класів); його поведінку можна поєднати із паттерном "Одинак, наприклад, щоб видаляти останній(-ні) введений(-ні) запис(-си)".
2. (2,5 бали) Побудувати відповідні UML-діаграми.

Зрозуміло, що студент повинен повністю розуміти код. Щодо UML-діаграм, то оцінка виставлятиметься, в першу чергу, за старання, а не за правильність побудови.

Підказки відповідно до варіантів

1. Паттерн “Одинак” можна застосувати для ведення журналу продажів комп'ютерної техніки. Паттерн “Фасад” можна застосувати для продажу одразу ігрового чи офісного, чи іншого виду комп'ютера. Паттерн “Хранитель” можна застосувати для відміни операції продажу.
2. Паттерн “Одинак” можна застосувати для ведення журналу наданих сервісних послуг. Паттерн “Фасад” можна застосувати для надання

одночасно низки послуг, що включає чистку, ремонт тощо. Паттерн “Хранитель” можна застосувати для відміни наданої послуги.

3. Паттерн “Одинак” можна застосувати для формування бази здійснених банківських операцій. Паттерн “Фасад” можна застосувати для видачі комплекту банківських карток на всі випадки життя. Паттерн “Хранитель” можна застосувати для відміни здійсненої банківської операції.
4. Паттерн “Одинак” можна застосувати для ведення реєстру присутніх на форумі людей. Паттерн “Фасад” можна застосувати для видачі комплекту виду: блокнот, ручка, бейджик, програми форуму. Паттерн “Хранитель” можна застосувати для відміни внесення людини у список.
5. Паттерн “Одинак” можна застосувати для формування реєстру виданих ліків за рецептом. Паттерн “Фасад” можна застосувати для формування комплекту першої домедичної допомоги. Паттерн “Хранитель” можна застосувати для відміни продажу лікарського засобу.
6. Паттерн “Одинак” можна застосувати для ведення журналу відвідувань. Паттерн “Фасад” можна застосувати для формування типового алгоритму стрижки. Паттерн “Хранитель” можна застосувати для відміни помилкового внесення клієнта.
7. Паттерн “Одинак” можна застосувати для формування реєстру клієнтів. Паттерн “Фасад” можна застосувати для автоматизації приготування піци поваром. Паттерн “Хранитель” можна застосувати для відміни внесення клієнта в реєстр відвідувачів.
8. Паттерн “Одинак” можна застосувати для формування реєстру відвідувачів. Паттерн “Фасад” можна застосувати для автоматизації процесу руху бібліотекаря при пошуку старовинної книги в архіві. Паттерн “Хранитель” можна застосувати для відміни внесення відвідувача у реєстр.
9. Паттерн “Одинак” можна застосувати для формування реєстру відвідувачів. Паттерн “Фасад” можна застосувати для формування типової документизованої процедури видачі тварини. Паттерн “Хранитель” можна застосувати для відміни внесення особи в реєстр відвідувачів.
10. Паттерн “Одинак” можна застосувати для формування реєстру відгуків від клієнтів. Паттерн “Фасад” можна застосувати для формування

типового процесу організації роботи над проектом. Паттерн “Хранитель” можна застосувати для відміни внесення відгуку від клієнта.

11. Паттерн “Одинак” можна застосувати для формування реєстру бракованих іграшок. Паттерн “Фасад” можна застосувати для формування алгоритму покомпонентної побудови іграшки. Паттерн “Хранитель” можна застосувати для відміни дії внесення іграшки в реєстр браку.
12. Паттерн “Одинак” можна застосувати для формування списку проданого. Паттерн “Фасад” можна застосувати для формування комплектів виду: для ванної, кухонний набір тощо. Паттерн “Хранитель” можна застосувати для відміни додавання запису у список проданого.
13. Паттерн “Одинак” можна застосувати для формування списку відвідувачів. Паттерн “Фасад” можна застосувати для вказання типових акупунктурних послідовностей вколювання. Паттерн “Хранитель” можна застосувати для викреслення доданого відвідувача зі списку відвідувачів.
14. Паттерн “Одинак” можна застосувати для формування списку бракованих електродеталей. Паттерн “Фасад” можна застосувати для вказання типових комплектів електродеталей на якийсь із конкретних випадків життя. Паттерн “Хранитель” можна застосувати для вилучення помилково внесеної електродеталі зі списку бракованих.
15. Паттерн “Одинак” можна застосувати для формування списку клієнтів. Паттерн “Фасад” можна застосувати для формування комплектів спеціальних товарів виду: для стрижки, для миття голови, для комфортного перебування клієнтів тощо. Паттерн “Хранитель” можна застосувати для вилучення помилково доданого клієнта.
16. Паттерн “Одинак” можна застосувати для формування списку клієнтів. Паттерн “Фасад” можна застосувати для формування “поличок” з типовими наборами для студента, вегетаріанця тощо. Паттерн “Хранитель” можна застосувати для вилучення помилково доданого клієнта.
17. Паттерн “Одинак” можна застосувати для формування реєстру імен клієнтів. Паттерн “Фасад” можна застосувати для формування

алгоритмів формування книг у твердій обкладинці з офсетним папером всередині, у глянцевій обкладинці зі звичайним папером всередині тощо. Паттерн “Хранитель” можна застосувати для відміни додавання клієнта у реєстр.

18. Паттерн “Одинак” можна застосувати для формування реєстру клієнтів. Паттерн “Фасад” можна застосувати для формування “пачок” документів на здійснення операції купівлі-продажу квартири, машини тощо. Паттерн “Хранитель” можна застосувати для відміни додавання клієнта у реєстр.
19. Паттерн “Одинак” можна застосувати для формування реєстру клієнтів. Паттерн “Фасад” можна застосувати для формування комплектів одягу для спорту, туризму тощо. Паттерн “Хранитель” можна застосувати для відміни додавання клієнта у реєстр.
20. Паттерн “Одинак” можна застосувати для формування реєстру клієнтів. Використати паттерн “Фасад”. Підказка: можна, щоб турагентство одразу й організовувало забезпечення своїх клієнтів комплектами одягу та спорядження для різного виду туристичного “відпочинку”. Це, наприклад, можуть бути: комплект теплого одягу, лижі з каскою, палатка, харчування тощо. Паттерн “Хранитель” можна застосувати для відміни додавання клієнта у реєстр.
21. Паттерн “Одинак” можна застосувати для формування реєстру клієнтів. Використати паттерн “Фасад”. Підказка: можна, щоб страхова компанія видавала окрім, наприклад, страхування здоров’я чи квартири, чи машини, чи будь-чого іншого (погуглите), ще й здоров’я+квартиру чи здоров’я+машину, чи все це разом зі знижками. Паттерн “Хранитель” можна застосувати для відміни додавання клієнта у реєстр.
22. Паттерн “Одинак” можна застосувати для формування списку прибувших водіїв. Паттерн “Фасад” можна застосувати для формування типового комплексу дій для виїзду водія з парковки. Паттерн “Хранитель” можна застосувати для відміни додавання водія у список.

Коротка характеристика паттернів

Породжуючі

Одинак – якщо щось має стати єдиним у програмі: адміністратор, база даних тощо.
Прототип – якщо потрібно створити точну копію деякого об’єкта (не виключено, що, при цьому, вноситиметься певна невелика зміна значень параметрів методу).
Будівельник – якщо необхідно створити шаблон із послідовним виконанням низки методів.
Фабричний метод – якщо потрібно, щоб повернувся той чи інший клас, залежно від тих чи інших умов.
Абстрактна фабрика – якщо в програмі є кілька фабричних методів, то варто об'єднати їх в єдину фабрику.

Структурні

Фасад – якщо має виконатись певний перелік методів залежно від тієї чи іншої ситуації.
Міст – якщо деякий функціонал потрібно розділити на 2 класи з метою уникнення дубляжу коду.
Легковаговик – якщо у величезному масиві об'єктів часто зустрічаються поля з однаковими ресурсоємним значеннями.
Проксі (заступник) – якщо потрібно створити клас, який би керував іншим класом.
Адаптер – якщо потрібно адаптувати деякий існуючий функціонал до нового.
Компонувальник – щось там з бінарними деревами та рекурсією.
Декоратор – якщо існуючий функціонал певного класу потрібно доповнити новим.

Поведінкові

Хранитель – якщо потрібно зберігати/відновлювати попередній(-ні) стан(-ни) об'єкта(-ів).

Рекомендована література

1. Сайт https://metanit.com/sharp/tutorial/.
2. Сайт https://refactoring.guru/uk/design-patterns:
    - https://refactoring.guru/uk/design-patterns/catalog
    - https://refactoring.guru/uk/design-patterns/csharp

3. Книга https://learn.ztu.edu.ua/pluginfile.php/632/mod_resource/content/1/DesignPatterns_AndriyBuday.pdf.
4. Текстовий лекційний матеріал.

Додаткова література

1. https://www.youtube.com/watch?v=dhnsegiPXoo&list=PLLWMQd6PeGY3ob0Ga6vn1czFZfW6e-FLr
2. https://www.youtube.com/watch?v=VqgXn7wsPsc&list=PLuGqgO5WmeGOGGbaBwJvvQ7_Su6cIBc8C