**Опис підсистем проекту "Магазин" та використовуваних шаблонів**

Проект має наступні підсистеми:

1. **Зал магазину (Shop floor)** – підсистема, яка виконує процес обслуговування клієнтів, має декілька кас та може симулювати черги, також має список виставлених товарів зі складу для продажу та може поповнюватись шляхом переносу товарів зі складу. Реалізує **шаблон одинака (Singleton)**, тому що кожен магазин повинен мати тільки один торговий зал (якщо залів декілька, вони вважаються за один, так як процес обслуговування клієнтів в них спільний). Також для процесу симулювання кас можна використати шаблон **стратегії (Strategy)**, але не обов’язково, так як стратегію можна відразу задати стабільною, бо вона застосовується тільки для клієнтів магазину.
2. **Склад (Storage)** – база товарів на майбутнє, яка призначена для поповнення товарів у залі магазину та може поповнюватись шляхом закупівлі або створення власними силами. Можливо обійтись без складу, якщо його функції передати залу магазину, тобто вважати, що усі товари відразу попадають у магазин для продажу. Склад є клієнтом декількох фабрик, які виконують функцію поповнення товарів та реалізують **шаблон абстрактної фабрики (Abstract fabric)**. Це дозволяє у будь-який момент легко змінити або доповнити список постачальників/виробників, які співпрацюють з магазином. Також склад має відкриті методи перегляду товарів (вибору та відповідного сортування) за різними ознаками.
3. **Офіс адміністратору (Administrative office)** – виконує функції організації процесу, а саме, відкриття/закриття магазину, відкриття/закриття окремих кас та запуск процесу продажу товарів, відстежування подій, реагування на винятки, здійснення закупівель (виготовлення товарів), якщо вони закінчуються на складі або вийшов їх строк придатності (у цьому разі товар виключається зі складу). Адміністратор також може вручну коригувати помилки у списку товарів, встановлювати та змінювати ціни. Офіс може працювати як автономно (без адміністратора-користувача), тобто за встановленими в програмі правилами, так і за ручними діями адміністратора. Він також реалізує взаємодію між різними типами користувачів, які можуть бути присутні у системі одночасно, і для цього реалізується шаблон **посередника (Mediator)**: відстежування подій та винятків може бути реалізовано саме через цей шаблон.
4. **Інтерфейс взаємодії з користувачем (Menu subsystem)** – містить різні рівні команд для різних типів користувачів. **Адміністратор** може виконувати усі функції офісу адміністратора, а також усі функції модератора і звичайного користувача; **модератор** може проглядати лог-файл і реагувати на події та винятки та виконувати функцію звичайного користувача; **користувач (клієнт/покупець)** може тільки переглядати товари у залі магазину та покупати товари за допомогою «корзини», і якщо при покупці виявиться застарілий товар, то він повідомляє про це модератора. Звісно, ця підсистема може реалізувати **шаблон** **фасаду (Facade)**, який скриватиме велику частину дій від класу Program. Замічу, що поки що не передбачено мати продавців та консультантів у залі магазину, система запланована як така, що може саморегулюватись, як універсам або інтернет-магазин без допоміжного персоналу.
5. **Підсистема логірування (Logging subsystem)** – веде запис помилок, подій, винятків, входу та виходу користувачів, процесу обслуговування клієнтів, закупівлі товарів та ліквідації товарів з вичерпаним строком придатності. Може відповідати на запити від адміністратора та модератора.
6. **Сервісна підсистема (Low level service subsystem)** – містить класи і методи низькорівневої взаємодії з файлами та екраном, тобто методи, які можуть сильно змінюватись під час розвитку проекту. Потрібно реалізувати вивід та читання лог-файлу, збереження стану (серіалізацію) складу та залу магазину на поточний момент на випадок тимчасового закриття магазину, з відновленням (десеріалізацією) при наступному відкритті, а також доступ до бази користувачів (їх імен та паролів).
7. **Підсистема входу в систему (Authenticating subsystem) –** має базу користувачів та підтримує їх реєстрацію, вхід/вихід з системи, перевірку їх прав та надання доступу відповідного рівня.
8. **Для проектування дерева ієрархії класів продуктів** можна використати як окремі інтерфейси з різними полями для різних типів продуктів, так і **паттерн декоратора (Decorator).** Це дає змогу як «будувати» різні продукти (наприклад, різні варіанти піци, торту або кави), або обирати зі списку товарів продукти з різними характеристиками. Але ці характеристики повинні бути відомі заздалегідь. Наприклад, адміністратор не може змінити характеристики продуктів, які має в своєму розпорядженні, він може (і повинен) тільки вказати ці характеристики при введенні продуктів.