**Лабораторна робота №4**

**«Застосування патернів програмування»**

**Завдання:**

1. Аналізуючи діаграми класів (Лабораторна робота №3), обґрунтувати застосування підібраних для реалізації патернів.

**Обґрунтування застосування шаблонів у системі управління фермою.**

**1. Singleton — для класів, які взаємодіють із БД**

**Де застосовано:** AnimalRepository, FeedRepository, AppointmentRepository

**Обґрунтування:**  
Хочу, щоб був **єдиний екземпляр** класу доступу до бази даних для кожного типу сутності, що уникне зайвого підключення, зменшить навантаження та забезпечить узгодженість даних.

**2. Factory Method — для створення тварин, кормів, оглядів**

**Де застосовано:** у AnimalService.createAnimal(), FeedService.createFeed(), AppointmentService.createAppointment()

**Обґрунтування:**  
Це дозволяє інкапсулювати логіку створення об’єктів з валідацією. Можемо легко розширити створення для різних видів тварин, типів корму тощо.

**3. Observer — сповіщення про події (низький рівень корму, новий огляд)**

**Де застосовано:** FeedService → NotificationService  
AppointmentService → NotificationService

**Обґрунтування:**  
Коли відбувається якась зміна стану (наприклад, критичний рівень корму або створено огляд), система автоматично **сповіщає** відповідальних осіб, не змінюючи основну бізнес-логіку.

**4. Strategy — для різних політик сортування або фільтрації (наприклад, кормів або тварин)**

**Де застосовано:** AnimalService.sortAnimals(strategy)  
FeedService.sortFeeds(strategy)

**Обґрунтування:**  
Дозволяє змінювати **алгоритм сортування** без зміни класів, наприклад: сортування за віком, видом, станом здоров’я.

**5. Facade — для спрощеного інтерфейсу взаємодії**

**Де може бути застосовано:** FarmManagementFacade — об’єднує AnimalService, FeedService, AppointmentService

**Обґрунтування:**  
Зовнішній користувач (адміністратор або фермер) працює з єдиним фасадом замість того, щоб викликати сервіси окремо. Це спрощує інтеграцію та тестування.

**Узагальнення:**

* Підібрані патерни логічно вписуються у структуру класів.
* Вони підвищують **розширюваність**, **гнучкість**, **реюзабельність** коду.
* При потребі — легко додати **Decorator**, **Command** чи інші патерни (наприклад, для undo/redo чи дій користувача).

2. Використовуючи лабораторні роботи із дисциплін професійної підготовки, створити репозиторій, розмістити файли проектів. Надати доступ викладачам на github / gitlab ресурсі.

Посилання на репозиторій в github:  
<https://github.com/MaximilianMMM/Projectuvania_programnih_system>