Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Технології розроблення програмного забезпечення»»

Тема: «Вступ до паттернів проектування»

Виконала:

Студентка групи ІА-31

Соколова Поліна

Київ – 2025

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

15. E-mail клієнт

E-mail клієнт - це програма для управління електронною поштою, яка повинна забезпечувати:

- Підтримку протоколів POP3, SMTP, IMAP
- Автоналаштування для українських провайдерів (Gmail, ukr.net, i.ua)
- Організацію повідомлень у папки/категорії/важливість
- Роботу з чернетками
- Управління прикріпленнями до повідомлень

Патерни проектування для виконання:
singleton, builder, decorator, template method, interpreter, client-server.

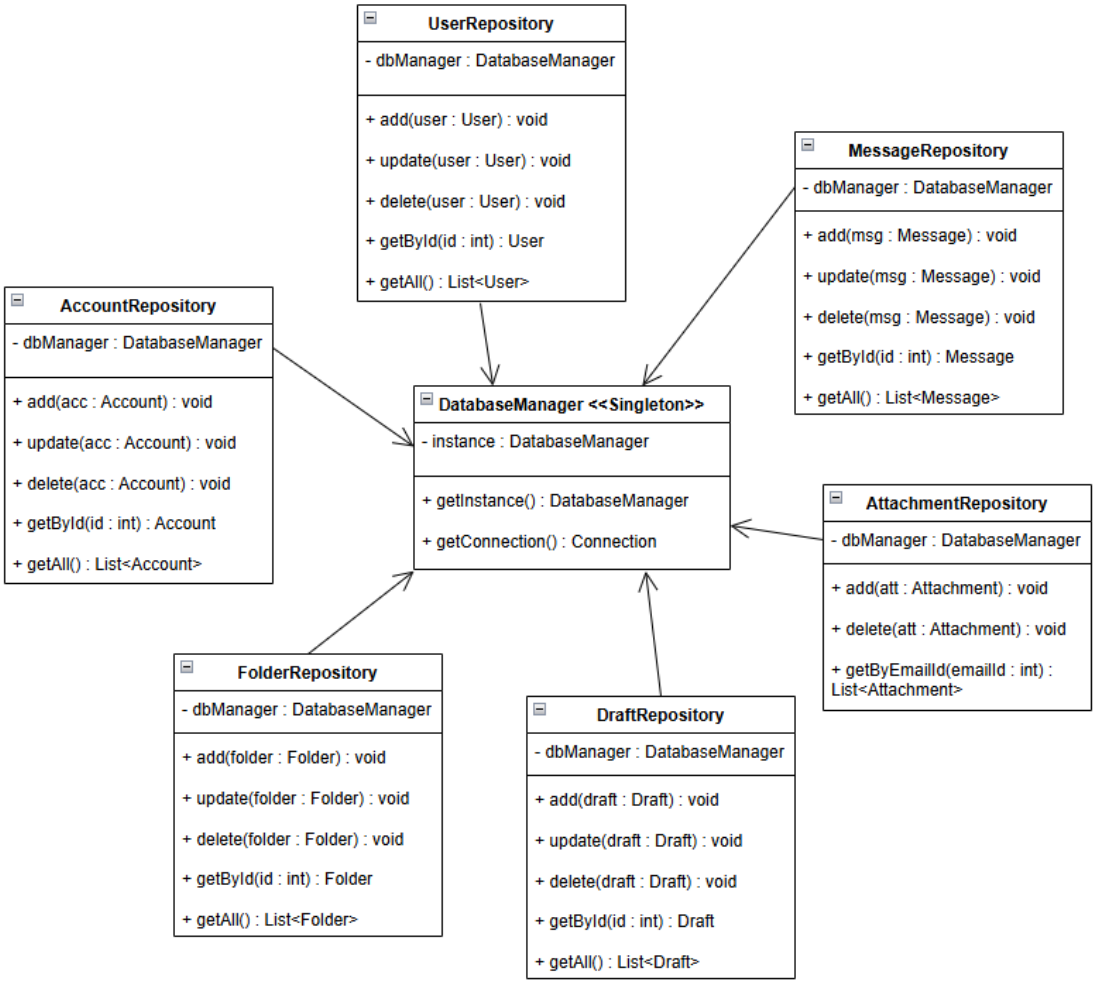

Рис.1 Діаграма класів, яка представляє використання шаблону «Singleton»

1

```
public class DatabaseManager {  16 usages
    private static DatabaseManager instance;  3 usages
    private static final String URL = "jdbc:sqlite:emailclient.db";  1 usage

    private DatabaseManager() {  1 usage
    }
    public static synchronized DatabaseManager getInstance() {  no usages
        if (instance == null) {
            instance = new DatabaseManager();
        }
        return instance;
    }
    public Connection getConnection() throws SQLException {  30 usages
        return DriverManager.getConnection(URL);
    }
}
```

2

```
public class UserRepository implements Repository<User> {  3 usages
    private final DatabaseManager dbManager = DatabaseManager.getInstance();
```

Рис.2 Фрагменти коду по реалізації шаблону

(1- клас DatabaseManager, 2 – UserRepository)

Висновок: у даній лабораторній роботі було реалізовано шаблон Singleton у межах проєкту E-mail клієнта**.** Клас DatabaseManager тепер має лише один екземпляр, який відповідає за підключення до бази даних. Усі класи, що працюють з БД (UserRepository, AccountRepository і т.д.), тепер отримують екземпляр DatabaseManager через getInstance(). Використання шаблону дозволило забезпечити єдину точку доступу до ресурсів, уникнути дублювання підключень, оптимізувати використання системних ресурсів.

Відповіді на питання:

1. Що таке шаблон проєктування?

Шаблон проєктування - це формалізований опис задачі, яка часто зустрічається при проєктуванні, її вдале рішення та рекомендації по застосуванню цього рішення в різних ситуаціях. Це "ескізи" архітектурних рішень, які зручно застосовувати у відповідних обставинах.

2. Навіщо використовувати шаблони проєктування?

Модель системи, побудована в межах патернів, є структурованим виокремленням значимих елементів і зв'язків, вона більш проста і наочна у вивченні, при цьому дозволяє глибоко опрацювати архітектуру системи. Застосування правильних патернів підвищує стійкість системи до зміни вимог та спрощує подальше доопрацювання.

3. Яке призначення шаблону «Стратегія»?

Шаблон Strategy дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом. Він зручний у випадках, коли існують різні "політики" обробки даних. Наприклад, це може бути використано для різних алгоритмів сортування.

5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

До шаблону входять Context (контекст, що містить посилання на стратегію), інтерфейс Strategy (загальний інтерфейс для всіх алгоритмів) та ConcreteStrategy (конкретні реалізації алгоритмів). Контекст містить поле strategy з посиланням на об'єкт стратегії і делегує йому виконання алгоритму. Коли стратегію потрібно замінити, просто замінюється об'єкт в полі Context.strategy.

6. Яке призначення шаблону «Стан»?

Шаблон State дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Пов'язані зі станом поля, методи і дії виносяться в окремий загальний інтерфейс, а кожен стан являє собою окремий клас, який реалізує цей інтерфейс.

8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

Шаблон включає Context (об'єкт, що має стан), інтерфейс State (загальний інтерфейс для всіх станів) та ConcreteState (конкретні стани з різною поведінкою). Об'єкти, що мають стан, при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта. Це дозволяє легко додавати нові

стани, відокремлювати залежні від стану елементи в інших об'єктах, і відкрито проводити заміну стану.

9. Яке призначення шаблону «Ітератор»?

Iterator - це шаблон реалізації об'єкта доступу до набору елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції з самої колекції, досягаючи розподілу обов'язків: колекція відповідає за зберігання даних, ітератор - за прохід по колекції. Алгоритм ітератора може змінюватися, що дозволяє реалізовувати різноманітні способи проходження по колекції незалежно від виду і способу представлення даних.

11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

Шаблон містить інтерфейси Aggregate (колекція) та Iterator (ітератор), а також їх конкретні реалізації ConcreteAggregate та ConcreteIterator. Ітератор містить методи First() для установки покажчика на перший елемент, Next() для переходу до наступного, IsDone (булевське поле, яке встановлюється як true коли досягнуто кінця) та CurrentItem для отримання поточного об'єкта.

12. В чому полягає ідея шаблона «Одинак»?

Singleton - це клас, який може мати не більше одного об'єкта. Об'єкт зберігається як статичне поле в самому класі. Використання виправдано коли може бути не більше N фізичних об'єктів або необхідно жорстко контролювати всі операції через даний клас.

13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

Через те, що одинаки представляють собою глобальні дані, які мають стан. Стан глобальних об'єктів важко відслідковувати і підтримувати коректно, також їх важко тестувати. Вони вносять складність в програмний код через виклики в одне єдине місце з усіх ділянок коду, і при зміні підходу доведеться змінювати багато коду.

14. Яке призначення шаблону «Проксі»?

Proxy - це об'єкти-заглушки або двійники для об'єктів конкретного типу. Проксі об'єкти вносять додатковий функціонал або спрощують взаємодію з реальними об'єктами.

16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

Шаблон включає інтерфейс Subject, клас RealSubject (реальний об'єкт з основною логікою) та Proxy (проксі-об'єкт-замінник). Proxy і RealSubject реалізують один інтерфейс, а Proxy містить посилання на RealSubject. Клієнтський код працює з Proxy через інтерфейс, а проксі може додавати додаткову логіку (кешування, групування запитів), делегувати виклики RealSubject або контролювати доступ до нього.