

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: «Шаблони «Singleton»,  
«Iterator», «Proxy», «State»,  
«Strategy»»

Варіант №15

Виконав:  
студент групи ІА-23  
Лядський Д.С.

Перевірив:  
Мягкий М. Ю.

Київ 2024

## Зміст

Тема.....	3
Мета .....	3
Завдання .....	5
Обрана тема.....	6
Хід роботи .....	6
Завдання №2 .....	<b>Ошибка! Закладка не определена.</b>
Завдання №3 .....	<b>Ошибка! Закладка не определена.</b>
Завдання №4 .....	<b>Ошибка! Закладка не определена.</b>
Завдання №5 .....	<b>Ошибка! Закладка не определена.</b>
Структура існуючої частини проекту .....	<b>Ошибка! Закладка не определена.</b>
Висновок.....	8

## **Тема**

Шаблони «Singleton», «Iterator», «Proxy», «State», «Strategy»

## **Мета**

Дослідити принципи реалізації та використання шаблонів проектування «Singleton», «Iterator», «Proxy», «State» та «Strategy». Отримати практичні навички їх впровадження у програмному забезпеченні для підвищення його модульності, масштабованості та гнучкості. Аналізувати переваги та недоліки кожного з шаблонів у конкретних сценаріях.

## **Короткі теоретичні відомості**

Шаблони проектування — це повторювані рішення типових задач проектування зі сформульованими рекомендаціями. Вони спрощують розробку, роблять моделі зрозумілими й адаптивними, допомагають вибирати оптимальні рішення.

Застосування шаблонів проектування не гарантує, що розроблена архітектура буде кристально чистою і зручною з точки зору програмування. Однак в потрібних місцях застосування шаблонів дозволить досягти наступних вигод:

- Зменшення трудовитрат і часу на побудову архітектури;
- Надання проєктованій системі необхідних якостей (гнучкість, адаптованість, ін.);
- Зменшити накладні витрати на подальшу підтримку системи;
- Та інші.

Варто також зазначити, що знання шаблонів проектування допомагає не тільки архітекторам програмних систем, але і розробникам. Коли кожна людина в команді знає значення і властивості шаблонів, архітекторів простіше донести загальну ідею архітектури системи, а розробникам - простіше зрозуміти.

Оскільки, урешті-решт, кожен бізнес зводиться до грошей, шаблони проектування також є економічно виправданим вибором між побудовою власного «колеса», та реалізацією закріплених і гарантованих спільнотою розробників практик і підходів. Це звичайно ж не означає, що їх необхідно використовувати в кожному проекті на кожну вимогу. Підходи не є догмою, їх потрібно використовувати з головою.

### Шаблон «Singleton»

Призначення: гарантує наявність одного екземпляра класу з глобальною точкою доступу.

Приклади: налаштування програми, уряд країни.

Переваги:

- Контрольована кількість об'єктів.
- Простий доступ.

Недоліки:

- Ускладнює тестування.
- Може призводити до поганого дизайну.

### Шаблон «Iterator»

Призначення: забезпечує послідовний доступ до елементів колекції без розкриття її структури.

Приклади: віртуальний гід у місті.

Переваги:

- Універсальний доступ до даних.
- Простота в додаванні нових алгоритмів обходу.

Недоліки:

- Не завжди потрібен, якщо достатньо звичайного циклу.

### Шаблон «Proxy»

Призначення: представляє об'єкт-заступник для іншого об'єкта, додаючи проміжну логіку.

Приклади: банківська картка як заступник готівки.

Переваги:

- Контроль доступу до сервісу.
- Можливість роботи без створення основного об'єкта.

Недоліки:

- Ускладнення коду.
- Затримки у виконанні.

### Шаблон «State»

Призначення: змінює поведінку об'єкта залежно від його стану.

Приклади: тарифні плани хостингу, типи банківських карток.

Переваги:

- Легка обробка нових станів.
- Розмежування поведінки за станами.

Недоліки:

- Можливе ускладнення коду через велику кількість станів

## Завдання

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

## Обрана тема

### 15 E-mail клієнт (singleton, builder, decorator, template method, interpreter, SOA)

Поштовий клієнт повинен нагадувати функціонал поштових програм Mozilla Thunderbird, The Bat і т.д. Він повинен сприймати і коректно обробляти pop3/smtp/imap протоколи, мати функції автонастройки основних поштових провайдерів для України (gmail, ukr.net, i.ua), розділяти повідомлення на папки/категорії/важливість, зберігати чернетки незавершених повідомлень, прикріплювати і обробляти прикріплені файли.

### Хід роботи

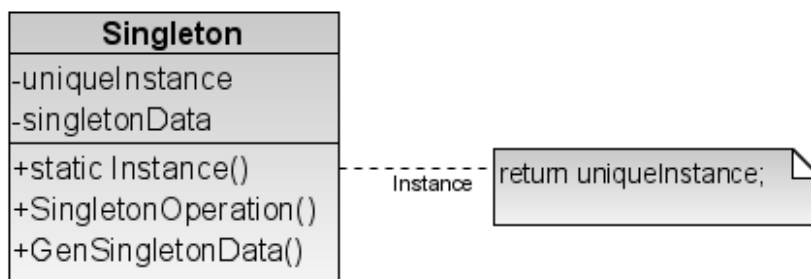


Рисунок 1. Діаграма класів, що описує структуру шаблону проектування Одинак

## Реалізації патерну в проєкті:

```

7  public class DbConnection {
8      private static volatile DbConnection dbConnection;
9      private final Connection connection;
10     private static final String URL = "jdbc:postgresql://127.0.0.1:5432/E-mail client";
11     private static final String USER = "postgres";
12     private static final String PASSWORD = "admin";
13
14     private DbConnection() { this.connection = connect(); }
15
16     public static DbConnection getInstance(){
17         if (dbConnection == null){
18             synchronized (DbConnection.class){
19                 if (dbConnection == null){
20                     dbConnection = new DbConnection();
21                 }
22             }
23         }
24         return dbConnection;
25     }
26 }

```

Рисунок 2. Реалізації патерну в проєкті

Цей код представляє клас DbConnection, який реалізує шаблон одинички (Singleton) для роботи з базою даних PostgreSQL. Це забезпечує створення єдиного екземпляра з'єднання з базою даних для використання у всьому додатку.

### Опис ключових елементів:

`private static volatile DbConnection dbConnection` - Це статична змінна, яка зберігає єдиний екземпляр класу. Ключове слово `volatile` гарантує коректне оновлення значення змінної в багатопотоковому середовищі.

`private final Connection connection` - Зберігає об'єкт з'єднання з базою даних.

`private DbConnection()` - Приватний конструктор для заборони створення об'єктів цього класу ззовні. Викликає метод `connect()`, щоб встановити з'єднання з базою даних.

`public static DbConnection getInstance()` - Метод для отримання єдиного екземпляра класу. Реалізовано подвійне блокування (double-checked locking), щоб уникнути зайвого синхронізування після створення екземпляра.

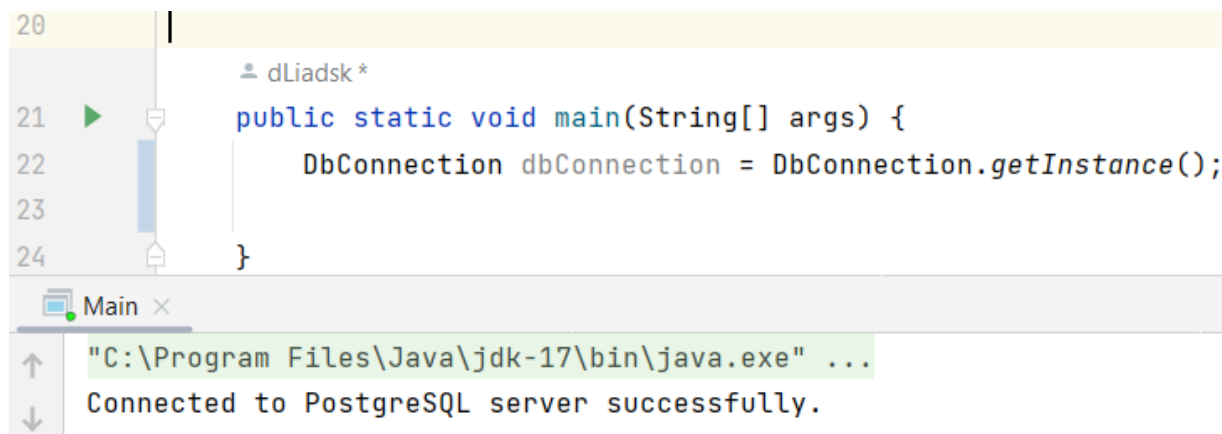
`private static Connection connect()` - Метод для встановлення з'єднання з базою даних.

Як працює код:

Перший виклик `DbConnection.getInstance()` створює екземпляр класу, якщо він ще не існує. Усі наступні виклики повертатимуть вже існуючий екземпляр. Під час створення екземпляра викликається метод `connect()`, що намагається встановити з'єднання з PostgreSQL.

Переваги:

- Ефективність: Об'єкт створюється тільки один раз.
- Потокобезпека: Використання `synchronized` і `volatile` забезпечує коректну роботу в багатопотоковому середовищі.
- Зручність: Єдине місце для роботи з підключенням до бази даних.



```

20 |
21 | public static void main(String[] args) {
22 |     DbConnection dbConnection = DbConnection.getInstance();
23 |
24 | }
  
```

Main x

"C:\Program Files\Java\jdk-17\bin\java.exe" ...

Connected to PostgreSQL server successfully.

Рисунок 3. Виклик методу для отримання єдиного екземпляра класу

## Висновок

У ході виконання лабораторної роботи було розглянуто та реалізовано шаблони проектування «Singleton», «Iterator», «Proxy», «State» та «Strategy».