Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота №7**

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Шаблон «Mediator», «Facade», «Bridge», «Template method»»

Варіант №15

Виконав: Перевірив:

студент групи ІА-23 Мягкий М. Ю.

Лядський Д.С.

Київ 2024

Зміст

[Тема 3](#_Toc184220812)

[Мета 3](#_Toc184220813)

[Короткі теоретичні відомості 3](#_Toc184220814)

[Завдання 5](#_Toc184220815)

[Обрана тема 6](#_Toc184220816)

[Хід роботи 6](#_Toc184220817)

[Висновок 10](#_Toc184220818)

# Тема

Шаблон «Mediator», «Facade», «Bridge», «Template method»

# Мета

Метою цієї лабораторної роботи є дослідження та застосування шаблонів проектування «Mediator», «Facade», «Bridge» і «Template Method». В рамках роботи студенти повинні розібратися в концепціях кожного шаблону, їхньому застосуванню для вирішення реальних задач, а також реалізувати приклади їх використання в коді.

# Короткі теоретичні відомості

Принципи проектування:

Don't Repeat Yourself (DRY):

Повторення коду слід уникати, щоб зробити його читабельним, легким у підтримці й уникнути помилок. Використовуються техніки рефакторингу, наприклад, винесення загального функціоналу в методи, інтерфейси або класи.

Keep It Simple, Stupid (KISS):

Компоненти системи мають бути простими. Краще створювати багато простих компонентів, ніж одну складну систему. Це підвищує надійність і спрощує підтримку.

You Only Load It Once (YOLO):

Ініціалізаційні змінні слід завантажувати один раз при запуску програми, щоб уникнути затримок і проблем з продуктивністю.

Принцип Парето (80/20):

80% результату досягається за 20% зусиль.

Наприклад, 80% помилок можна виправити, закривши 20% багів.

You Ain’t Gonna Need It (YAGNI):

Не варто реалізовувати функціонал, який, ймовірно, не знадобиться. Просте рішення краще, ніж складне і універсальне.

Шаблон «Mediator» (Посередник):

Призначення: Організовує взаємодію між об'єктами через окремий об'єкт-посередник, що зменшує кількість залежностей між компонентами.

Застосування: Використовується для керування складними взаємодіями, наприклад, в діалогах, де взаємодіють численні елементи.

Переваги:

* Усуває прямі залежності між компонентами.
* Централізує управління.

Недоліки:

* Посередник може стати занадто складним.

Шаблон «Facade» (Фасад):

Призначення: Надає спрощений інтерфейс доступу до складної системи.

Застосування: Використовується для ізоляції клієнта від деталей реалізації складної підсистеми, наприклад, бібліотек або фреймворків.

Переваги:

* Спрощує роботу з підсистемою.
* Зменшує кількість залежностей.

Недоліки:

* Фасад може стати божественним об'єктом.

Шаблон «Bridge» (Міст):

Призначення: Відокремлює абстракцію від її реалізації, дозволяючи розвивати їх незалежно.

Застосування:Використовується, коли потрібно уникнути множинного спадкування через перехресні комбінації властивостей, наприклад, кольору й форми.

Переваги:

* Забезпечує платформонезалежність.
* Реалізує принцип відкритості/закритості.

Недоліки:

* Ускладнює код через введення додаткових класів.

Шаблон «Template Method» (Шаблонний метод):

Призначення: Дозволяє визначити основну структуру алгоритму в базовому класі, залишивши підкласи для уточнення окремих його кроків.

Застосування: Використовується для створення повторюваних алгоритмів з варіативними кроками, наприклад, обробка файлів різних форматів.

Переваги:

* Усуває дублювання коду.
* Полегшує підтримку й розширення.

Недоліки:

* Може бути важко налаштувати для складних алгоритмів.

# Завдання

1. Ознайомитися з короткими теоретичними відомостями

2. Реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.

3. Застосування одного з даних 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

Реалізація патерну template method:



Рисунок 1. Абстрактний клас EmailProtocolHandler



Рисунок 2. Клас IncomingServerHandler

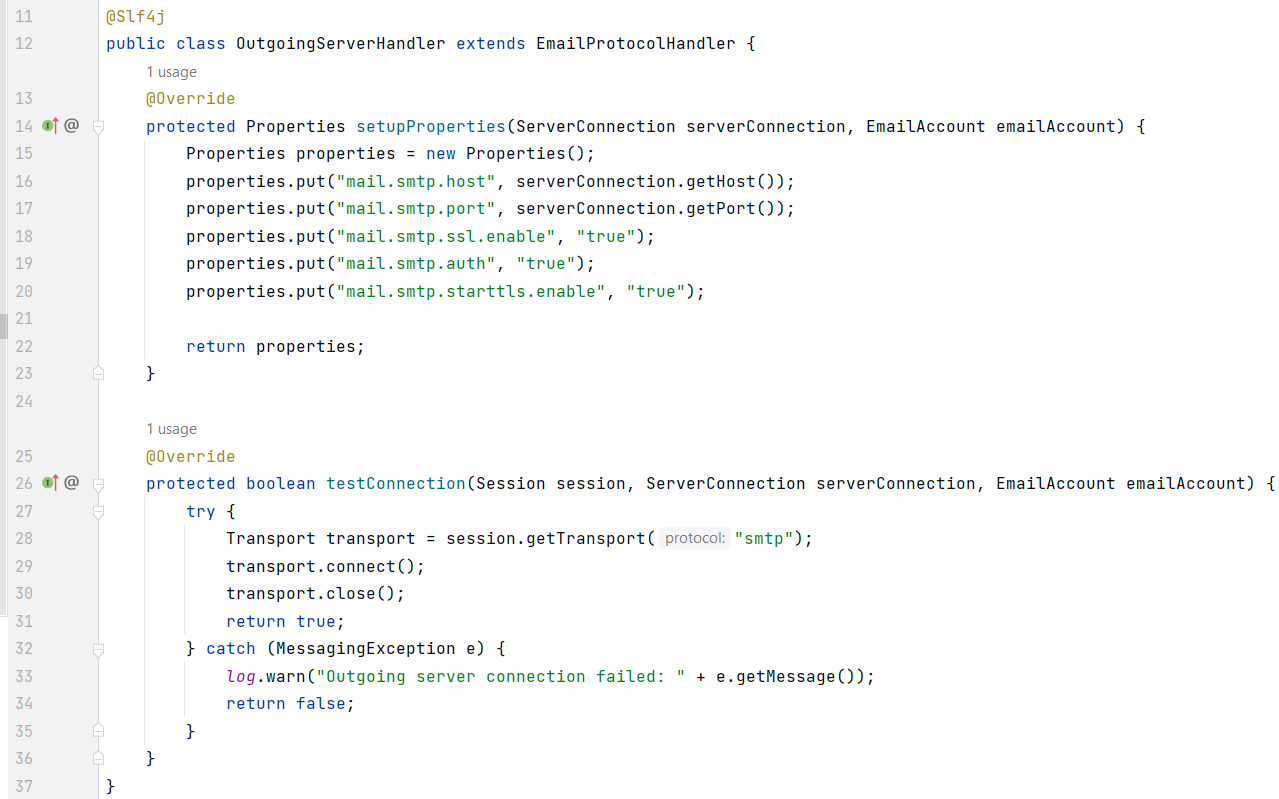


Рисунок 3. Клас EmailAccount

Даний код реалізує систему обробки протоколів електронної пошти для Java-додатка. Він розроблений на основі шаблонного методу, що дозволяє забезпечити загальну логіку авторизації для різних типів серверів (вхідних та вихідних), а специфічні аспекти реалізуються в дочірніх класах.

Опис класів:

1. EmailProtocolHandler

Цей абстрактний клас є базовим для роботи з протоколами електронної пошти. Він реалізує загальну логіку авторизації та передбачає методи для налаштування специфічних властивостей і тестування з'єднання, які повинні бути реалізовані в дочірніх класах.

Методи:

authorize: головний метод для створення сесії, що виконує:

* Налаштування властивостей через setupProperties.
* Створення Session.
* Перевірку з'єднання через testConnection.

setupProperties: абстрактний метод для налаштування властивостей залежно від протоколу.

testConnection: абстрактний метод для перевірки з'єднання.

createSession: приватний метод для створення сесії з використанням облікових даних.

2. IncomingServerHandler

Реалізує специфічну логіку для роботи з вхідними серверами (IMAP/POP3).

Особливості:

* Використовує протокол (IMAP або POP3), визначений у ServerConnection.
* Налаштовує SSL для безпечного з'єднання.
* Використовує Store для тестування з'єднання.

Методи:

setupProperties: налаштовує властивості для вхідного сервера (хост, порт, SSL).

testConnection: перевіряє з'єднання через Store.

3. OutgoingServerHandler

Реалізує специфічну логіку для роботи з вихідними серверами (SMTP).

Особливості:

* Використовує SMTP для відправлення повідомлень.
* Додає властивості для автентифікації та підтримки TLS.
* Використовує Transport для тестування з'єднання.

Методи:

setupProperties: налаштовує властивості для SMTP (хост, порт, автентифікація, SSL, TLS).

testConnection: перевіряє з'єднання через Transport.

Основні принципи проєктування:

Шаблонний метод (Template Method): Загальний алгоритм авторизації реалізований в базовому класі EmailProtocolHandler, тоді як специфічні кроки налаштування властивостей (setupProperties) і перевірки з'єднання (testConnection) делегуються дочірнім класам.

Полегшення розширення: Код легко розширити для інших протоколів (наприклад, EWS або протоколів майбутнього).

Логгування: Використовується @Slf4j для журналювання, що дозволяє відслідковувати успішні і невдалі спроби з'єднання.



Рисунок 4. Застосування в EmailAccountService



Рисунок 6. Демонстрація роботи

# Висновок

Під час виконання лабораторної роботи було успішно вивчено чотири популярних шаблони проектування: «Mediator», «Facade», «Bridge» і «Template Method». Застосування кожного з цих шаблонів дозволяє знизити складність програмних систем, зробити їх більш гнучкими, зрозумілими та масштабованими.