Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря
Сікорського” Факультет інформатики та
обчислювальної техніки

Лабораторна робота №7
«Технології розроблення програмного забезпечення»
Тема: « E-mail клієнт»

Виконав:
Студент групи
ІА-34
Марченко А.О.
Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Патерни проектування.

Мета: Вивчити структуру шаблонів «Mediator», «Facade», «Bridge», «Template method» та навчитися застосовувати їх в реалізації програмної системи.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

• Реалізувати один з розглянутих шаблонів за обраною темою.

• Реалізувати не менше 3-х класів відповідно до обраної теми.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

Теоретичні відомості

Шаблон «Mediator»

Призначення патерну: Шаблон «Mediator» (посередник) використовується для визначення взаємодії об'єктів за допомогою іншого об'єкта (замість зберігання посилань один на одного) . Даний шаблон схожий на шаблон «команда», проте в даному випадку замість зберігання даних про конкретну дію, зберігаються дані про взаємодії між компонентами.

Даний шаблон зручно застосовувати у випадках, коли безліч об'єктів взаємодіє між собою деяким структурованим чином, однак складним для розуміння. У такому випадку вся логіка взаємодії виноситься в окремий об'єкт. Кожен із взаємодіючих об'єктів зберігає посилання на об'єкт «медіатор».

Переваги та недоліки:

+ Організація взаємодії між об’єктами лише через посередника спрощує розуміння та супроводження такого коду.

+ Додавання нових посередників без зміни існуючих компонентів дозволяє розширювати систему без зміни існуючого коду.

+ Зменшення залежностей між об’єктами підвищує гнучкість системи.

- Посередник, з часом, може перетворитися на дуже складний об’єкт, який робить все («God Object»).

Шаблон «Facade»

Призначення патерну: Шаблон «Facade» (фасад) передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми. Оскільки підсистема може складатися з безлічі класів, а кількість її функцій – не більше десяти, то щоб уникнути створення «спагеті-коду» (коли все тісно пов'язано між собою) виділяють один загальний інтерфейс доступу, здатний правильним чином звертатися до внутрішніх деталей.

Це також відволікає користувачів від змін в підсистемі (внутрішня реалізація може змінюватися, а наданої послуги немає), що також скоротить кількість змін в використовуваних фасад класах (без фасаду довелося б змінювати вихідні коди в безлічі точок).

Переваги та недоліки:

+ Інкапсуляція внутрішньої структури від клієнтського коду.

+ Спрощується інтерфейс для роботи з модулем закритим фасадом.

- Зниження гнучкості в налаштуванні та використанні програмного коду закритого фасадом.

Шаблон «Bridge»

Призначення патерну: Шаблон «Bridge» (міст) використовується для поділу інтерфейсу і його реалізації. Це необхідно у випадках, коли може існувати кілька різних абстракцій, над якими можна проводити дії різними способами.

Переваги та недоліки:

+ Дозволяє змінювати ієрархії абстракції та реалізації незалежно одна від одної.

+ Розділивши абстракцію від реалізації отримуємо більшу гнучкість та простіший супровід такого коду.

- Підвищена гнучкість при використанні патерну отримується за рахунок більшої складності, введення додаткових проміжних рівнів.

Шаблон «Template Method»

Призначення патерну: Шаблон «Template Method» (шаблонний метод) дозволяє реалізувати покроково алгоритм в абстрактному класі, але залишити специфіку реалізації підкласам. Можна привести в приклад формування веб-сторінки: необхідно додати заголовки, вміст сторінки, файли, що додаються, і нижню частину сторінки. Код для додавання вмісту сторінки може бути абстрактним і реалізовуватися в різних класах – AspNetCompiler, HtmlCompiler, PhpCompiler і т.п. Додавання всіх інших елементів виконується за допомогою вихідного абстрактного класу з алгоритмом.

Переваги та недоліки:

+ Полегшує повторне використання коду.

- Ви жорстко обмежені скелетом існуючого алгоритму.

- Ви можете порушити принцип підстановки Барбари Лісков, змінюючи базову поведінку одного з кроків алгоритму через підклас.

- З ростом складності загального алгоритму шаблонний метод стає занадто складно підтримувати, особливо, коли є багато віртуальних методів для перевизначення в підкласах.

Хід роботи

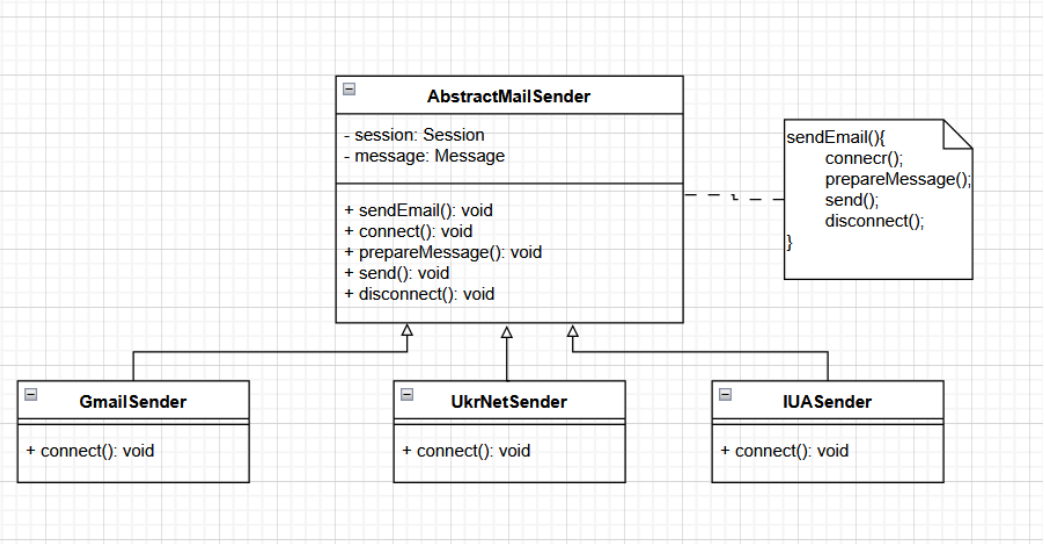


Рис. 7.1-Реалізація патерну Template Method

На діаграмі абстрактний клас AbstractMailSender містить загальні для всіх відправників поля session та message, а також визначає шаблонний метод sendEmail(), який задає стандартну послідовність дій: підключення, підготовку повідомлення, відправлення та від'єднання. Ці кроки описані як абстрактні методи, щоб підкласи могли реалізувати власну специфіку. Конкретні відправники — GmailSender, UkrNetSender та IUASender — успадковують базову поведінку і реалізують лише метод connect(), оскільки інші етапи можуть бути спільними або ще не змінюються у цьому варіанті. Таким чином, структура демонструє використання патерна «Шаблонний метод», де основний алгоритм визначено в абстракції, а конкретні реалізації відповідають лише за ті частини, що можуть відрізнятися.

Посилання на код:

https://github.com/BeautifulBublik/TRPZ/tree/master/EmailClient/src/main/java/com/example/emailclient/service/sender

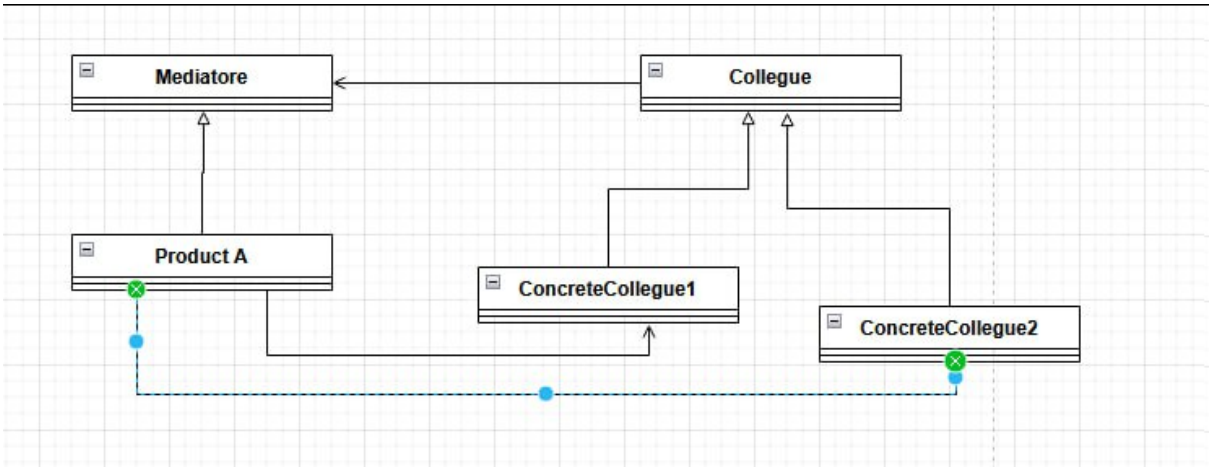
Питання до лабораторної роботи

1.Яке призначення шаблону «Посередник»?

Призначення шаблону «Посередник» – спрощувати взаємодію між численними об'єктами, роблячи їх менш зв'язаними, переміщуючи

комунікацію з об'єктів до одного класу-посередника . Це звільняє об'єкти від необхідності знати один про одного та дозволяє незалежно змінювати взаємодії між ними

2.Нарисуйте структуру шаблону «Посередник».



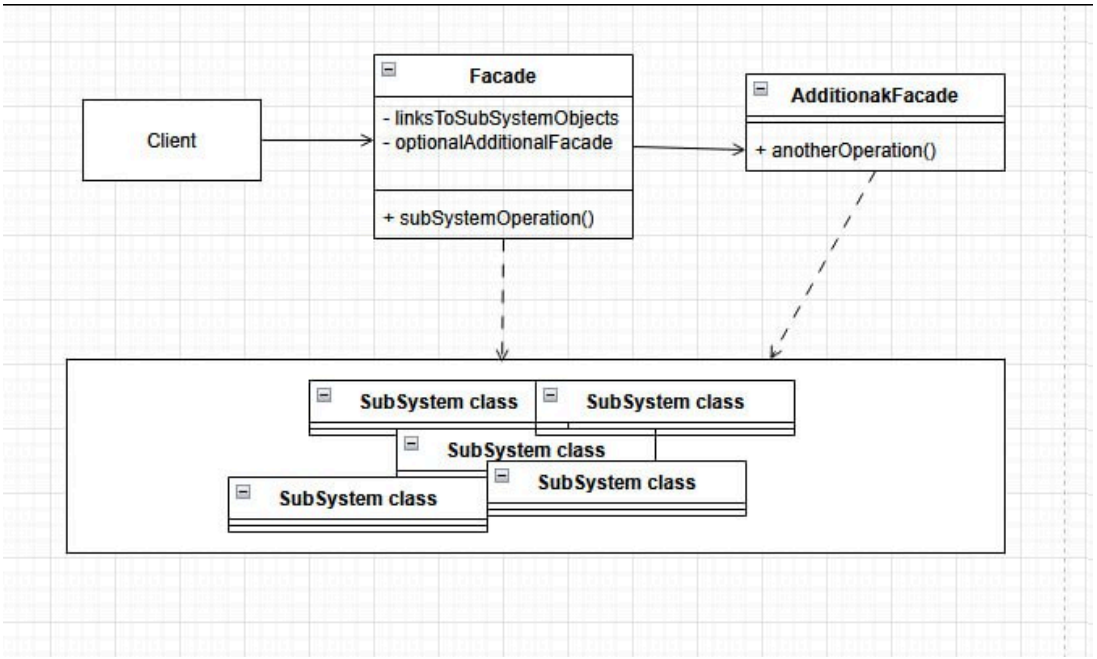
3.Які класи входять в шаблон «Посередник», та яка між ними взаємодія?

В шаблон "Посередник" (Mediator) входять класи: Посередник (Mediator), Конкретний Посередник (Concrete Mediator), Колег (Colleague) та Конкретний Колег (Concrete Colleague). Взаємодія відбувається між класом Посередник та класами Колег за допомогою повідомлень, де Посередник керує комунікацією між Колегами, а Колеги спілкуються виключно через Посередника, а не безпосередньо один з одним.

4.Яке призначення шаблону «Фасад»?

Шаблон «Facade» (фасад) передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми. Оскільки підсистема може складатися з безлічі класів, а кількість її функцій – не більше десяти, то щоб уникнути створення «спагеті-коду» виділяють один загальний інтерфейс доступу, здатний правильним чином звертатися до внутрішніх деталей.

5.Нарисуйте структуру шаблону «Фасад».



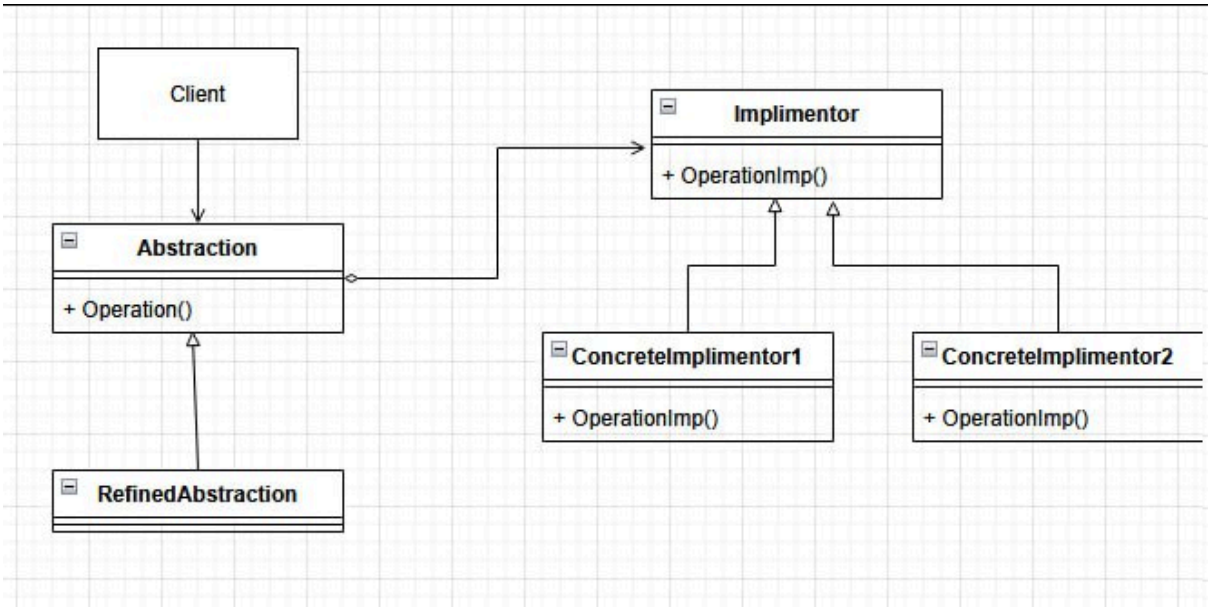
6.Які класи входять в шаблон «Фасад», та яка між ними взаємодія?

В шаблон "Фасад" входять Клієнт, Фасад та підсистеми (кілька класів). Клієнт взаємодіє тільки з Фасадом, не знаючи деталей роботи підсистем. Фасад делегує запити клієнта відповідним класам підсистеми

7.Яке призначення шаблону «Міст»?

Призначений для розділення абстракції та її реалізації, щоб їх можна було незалежно змінювати та розширювати. Це дозволяє відокремити клас, що визначає загальну поведінку , від класів, які реалізують цю поведінку на конкретних платформах. Таким чином, ви можете змінювати абстракцію або реалізацію, не впливаючи на іншу частину системи, що корисно для створення крос-платформних додатків або роботи з різними API.

8.Нарисуйте структуру шаблону «Міст».



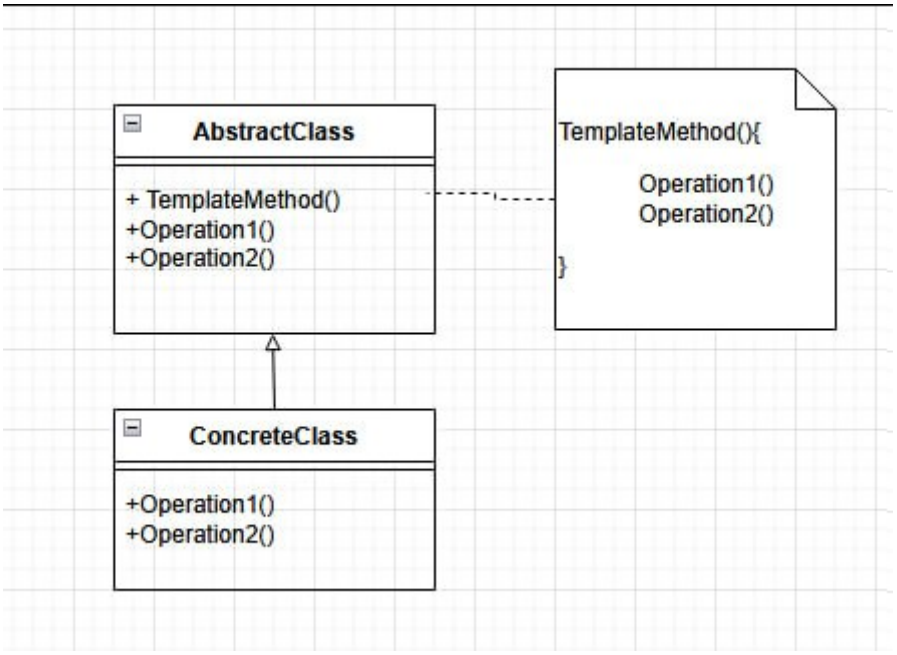
9.Які класи входять в шаблон «Міст», та яка між ними взаємодія?

Основні класи шаблону «Міст» Abstraction визначає загальний інтерфейс для клієнтів і підтримує посилання на об'єкт Реалізації (Implementor). Вона делегує основну роботу цьому об'єкту. Уточнена абстракція (Refined Abstraction) розширює інтерфейс Абстракції, додаючи специфічні функції або логіку. Реалізація (Implementor) визначає інтерфейс для конкретних (низькорівневих) операцій. Це може бути інтерфейс або абстрактний клас, який не залежить від Абстракції.

10.Яке призначення шаблону «Шаблоний метод»?

Призначення шаблону «Шаблоний метод» полягає у визначенні основи (каркасу) алгоритму в одному суперкласі та перекладанні відповідальності за певні його кроки на підкласи. Це дозволяє підкласам перевизначати окремі кроки, не змінюючи при цьому загальну структуру алгоритму.

11.Нарисуйте структуру шаблону «Шаблоний метод».



12.Які класи входять в шаблон «Шаблоний метод», та яка між ними взаємодія?

В шаблоні «Шаблоний метод» беруть участь абстрактний клас (AbstractClass), який визначає скелет алгоритму, та один або кілька конкретних класів (ConcreteClass), які реалізують окремі кроки цього алгоритму. Взаємодія відбувається через те, що абстрактний клас містить метод-шаблон, який визначає послідовність кроків алгоритму, викликаючи абстрактні або примітивні методи, а конкретні класи перевизначають ці примітивні методи для надання специфічної реалізаці.

13.Чим відрізняється шаблон «Шаблонний метод» від «Фабричного методу»?

Головна різниця між цими патернами полягає у їх призначенні: «Шаблонний метод» регламентує структуру алгоритму, дозволяючи підкласам змінювати окремі кроки, тоді як «Фабричний метод» регламентує створення об’єктів, передаючи підкласам відповідальність за вибір конкретного типу продукту. Якщо у «Шаблонному методі» ключове — це послідовність операцій, то у «Фабричному методі» — це процес створення відповідного об’єкта.

14.Яку функціональність додає шаблон «Міст»?

Патерн «Міст» додає можливість розділити абстракцію та її реалізацію, щоб вони могли змінюватися та розширюватися незалежно одна від одної.

Завдяки цьому зменшується кількість підкласів, послаблюється зв’язування між компонентами і з’являється гнучкість у виборі або зміні способу реалізації без зміни самої абстракції. Це дозволяє легко додавати нові варіанти поведінки та підтримувати код без дублювання.

Висновок: У цій роботі я використав шаблон Template Method, тому що процес надсилання електронного листа завжди має однакову загальну структуру — підключення, підготовку повідомлення, відправлення та від’єднання. Ця послідовність є фіксованою і описана один раз в абстрактному класі, щоб уникнути дублювання та забезпечити єдиний стандарт роботи. Проте конкретні кроки можуть відрізнятися залежно від поштового сервісу (Gmail, UkrNet, IUA), і саме ці змінні частини реалізуються у підкласах. Такий підхід дозволяє зберегти спільний алгоритм, але залишає можливість гнучко змінювати окремі його етапи, що і є головною ідеєю патерна Template Method. Також вивчив шаблони «Mediator», «Facade», «Bridge».