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

Лабораторна робота № 9

з дисципліни «Теорія розробки ПЗ»

Тема: РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE

Виконав:
студент групи ІА-14
Онуфрійчук Антон
Валерійович
Дата здачі _____
Захищено з балом _____

Перевірив: Мягкий Михайло
Юрійович

Київ, 2023

Тема: РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE

Завдання.

1. Ознайомитися з короткими теоретичними відомостями.

2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.

..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.

Для реалізації поштового клієнта з використанням принципів Сервіс-орієнтованої архітектури (SOA), ми маємо розділити функціональність застосунку на декілька незалежних сервісів. Кожен сервіс буде відповідати за певний аспект функціональності та зможе комунікувати з іншими сервісами через стандартизовані інтерфейси. Ось декілька ідей, як можна реалізувати SOA в розробці поштового клієнта:

1. Сервіси для Обробки Протоколів (POP3, SMTP, IMAP)

- Email Retrieval Service: Сервіс для отримання повідомлень, який підтримує POP3 та IMAP. Він відповідає за з'єднання з поштовим сервером та отримання повідомлень.

- Email Sending Service: Сервіс для відправлення електронних листів, який підтримує SMTP. Він відповідає за формування та відправлення електронних листів.

2. Сервіс Автонастройки Провайдерів

- Provider Configuration Service: Сервіс, який містить інформацію про налаштування основних поштових провайдерів та надає інтерфейс для швидкого налаштування клієнта під різні провайдери.

3. Сервіси для Управління Повідомленнями

- Message Sorting Service: Сервіс, який дозволяє сортувати, фільтрувати та категоризувати повідомлення на основі різних критеріїв (наприклад, відправник, тема).

- Draft Management Service: Сервіс для зберігання та управління чернетками незавершених повідомлень.

4. Сервіс Обробки Прикріплених Файлів

- Attachment Processing Service: Сервіс для завантаження, прикріплення та обробки файлів, що додаються до електронних листів.

5. Клієнтський Інтерфейс

- User Interface (UI) Service: Графічний інтерфейс користувача, який взаємодіє з вищезазначеними сервісами для надання користувачам доступу до функціональності поштового клієнта.

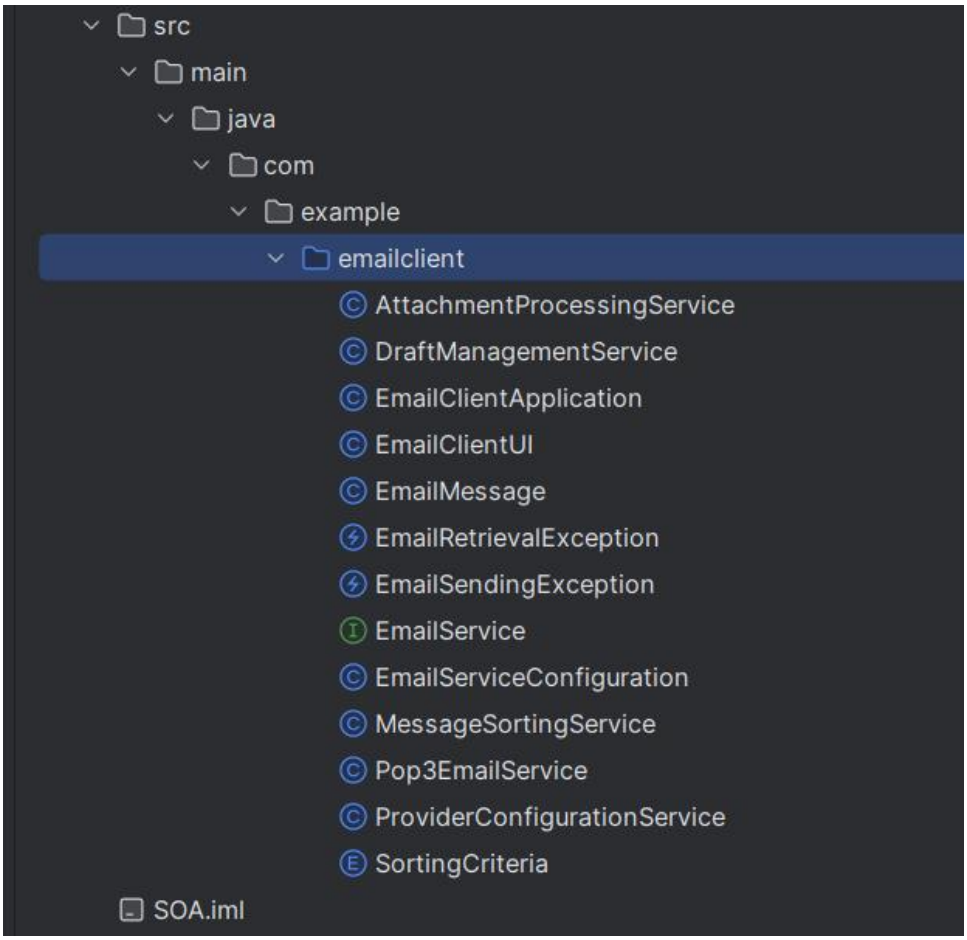
Взаємодія Сервісів

Кожен сервіс повинен мати визначений API для спілкування з іншими сервісами. Наприклад, UI Service буде викликати Email Retrieval Service та Email Sending Service для отримання та відправлення листів відповідно. Така модульна структура дозволяє легко змінювати, оновлювати та масштабувати кожен компонент системи незалежно від інших.

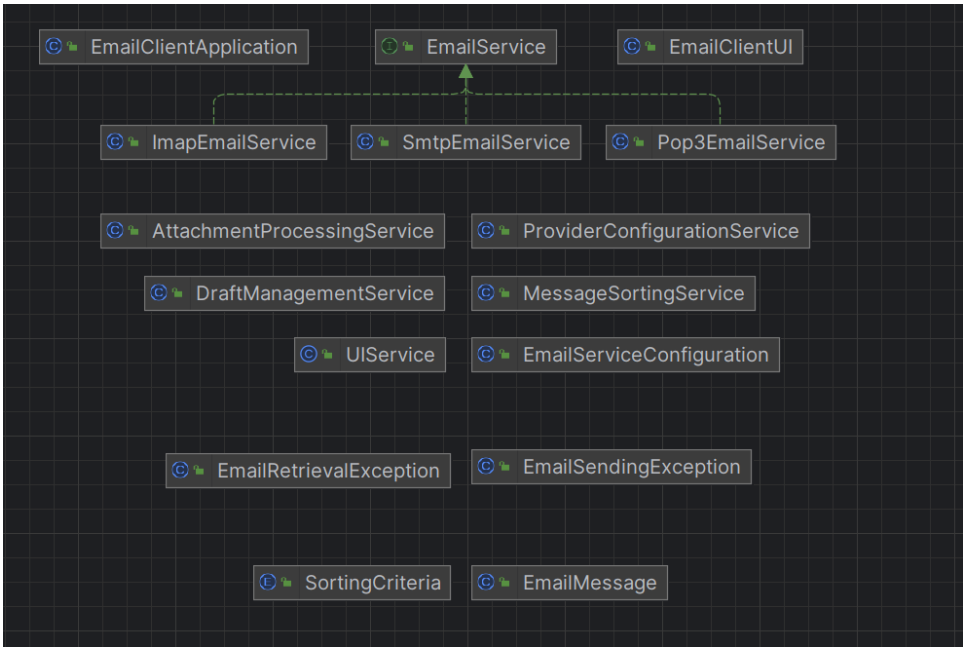
Масштабованість та Гнучкість

Використання SOA дозволяє легко масштабувати та адаптувати систему до змінних вимог, а також інтегрувати нові сервіси або замінювати існуючі без значних змін у інших частинах системи.

Структура проєкту:



Діаграма класів




```
public interface EmailService {
    void sendEmail(EmailMessage message) throws EmailSendingException;
```

```
    List<EmailMessage> retrieveEmails() throws EmailRetrievalException;
}

public class Pop3EmailService implements EmailService {
    @Override
    public void sendEmail(EmailMessage message) {
        // Код для відправлення електронного листа через SMTP
    }

    @Override
    public List<EmailMessage> retrieveEmails() {
        // Код для отримання електронних листів через POP3
        return new ArrayList<>();
    }
}

public class EmailMessage {
    private String from;
    private String to;
    private String subject;
    private String body;
    // Гетери та сетери
}

public class EmailSendingException extends Exception {
    // Конструктори, гетери
}

public class EmailRetrievalException extends Exception {
    // Конструктори, гетери
}

public class ProviderConfigurationService {
    public EmailServiceConfiguration getConfigurationForProvider(String providerName) {
```

```
        // Повертає конфігурацію для заданого провайдера
        return new EmailServiceConfiguration();
    }
}

public class EmailServiceConfiguration {
    private String smtpServer;
    private int smtpPort;
    private String pop3Server;
    private int pop3Port;
    // Гетери та сетери
}

public class MessageSortingService {
    public List<EmailMessage> sortEmails(List<EmailMessage> emails, SortingCriteria
criteria) {
        // Сортування повідомлень за заданим критерієм
        return new ArrayList<>();
    }
}

public class DraftManagementService {
    public void saveDraft(EmailMessage draft) {
        // Зберігання чернетки
    }

    public EmailMessage retrieveDraft(String draftId) {
        // Отримання чернетки
        return new EmailMessage();
    }
}

public enum SortingCriteria {
    DATE, SENDER, SUBJECT
```

```
}

public class AttachmentProcessingService {
    public void attachFile(EmailMessage emailMessage, File file) {
        // Додавання файлу до повідомлення
    }

    public List<File> extractAttachments(EmailMessage emailMessage) {
        // Отримання списку прикріплених файлів
        return new ArrayList<>();
    }
}

public class EmailClientUI {
    // Методи для взаємодії з користувачем (може включати GUI елементи)
}

public class EmailClientApplication {
    public static void main(String[] args) {
        // Ініціалізація та конфігурація сервісів
        // Взаємодія з EmailClientUI для демонстрації функціональності
    }
}

public class ImapEmailService implements EmailService {
    @Override
    public void sendEmail(EmailMessage message) {
        // Код для відправлення електронного листа через SMTP
    }

    @Override
    public List<EmailMessage> retrieveEmails() {
        // Код для отримання електронних листів через IMAP
        return new ArrayList<>();
```

```
    }
}

public class SmtpEmailService implements EmailService {
    @Override
    public void sendEmail(EmailMessage message) {
        // Код для відправлення електронного листа через SMTP
    }

    @Override
    public List<EmailMessage> retrieveEmails() {
        // Цей метод не використовується в SMTP
        throw new UnsupportedOperationException("SMTP does not support retrieving
emails.");
    }
}



public class UIService {

    private EmailService emailSendingService;
    private EmailService emailRetrievalService;

    public UIService(EmailService emailSendingService, EmailService
emailRetrievalService) {
        this.emailSendingService = emailSendingService;
        this.emailRetrievalService = emailRetrievalService;
    }

    public void sendEmail(EmailMessage emailMessage) {
        try {
            emailSendingService.sendEmail(emailMessage);
            System.out.println("Email sent successfully.");
```

```
        } catch (EmailSendingException e) {
            System.out.println("Failed to send email: " + e.getMessage());
        }
    }

    public void retrieveEmails() {
        try {
            List<EmailMessage> emails = emailRetrievalService.retrieveEmails();
            for (EmailMessage email : emails) {
                // Display emails to the user
            }
        } catch (EmailRetrievalException e) {
            System.out.println("Failed to retrieve emails: " + e.getMessage());
        }
    }

    // Методи для взаємодії з користувачем (можуть включати GUI елементи)
}
```

Висновок: При виконані лабораторної роботи імплементував SOA. Ця структура надає зразковий огляд різних класів та інтерфейсів, які можуть бути використані для розробки поштового клієнта з використанням принципів SOA.