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

Лабораторна робота №5

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проєктування»

Варіант: E-mail клієнт

Виконав:
студент групи ІА-32
Карповець Роман

Київ - 2025

Тема: Патерни проєктування.

Мета: Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

Тема проєкту: E-mail клієнт

Опис: 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.

Зміст

Теоретичні відомості

Призначення патерну: Шаблон «Builder» (Будівельник) використовується для відділення процесу створення об'єкту від його представлення. Це доречно у випадках, коли об'єкт має складний процес створення (наприклад, Web-сторінка як елемент повної відповіді web- сервера) або коли об'єкт повинен мати декілька різних форм створення (наприклад, при конвертації тексту з формату у формат).

Проблема: Візьмемо процес побудови відповіді на запит web-сервера. Побудова складається з наступних частин: додавання стандартних заголовків (дата/час, ім'я сервера, інш.), код статусу (після пошуку відповідної сторінки насервері), заголовки відповіді (тип вмісту, інш.), утримуване, інше.

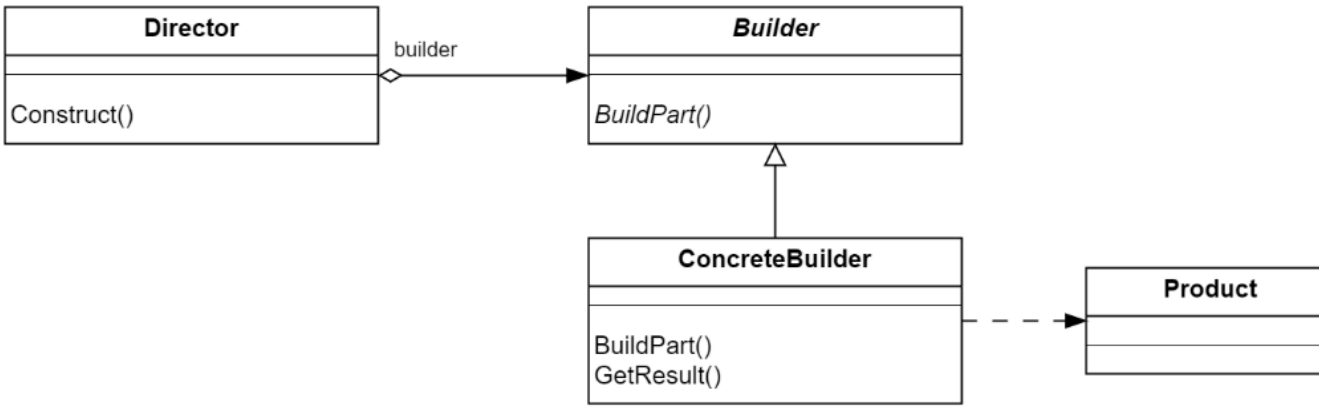

Рис 1. Структура патерну «Builder»

Рішення: Кожен з цих етапів може бути абстрагований в окремий метод будівельника. Це дасть наступні вигоди:

- Гнучкіший контроль над процесом створення сторінки;
- Незалежність від внутрішніх змін – наприклад, зміна назви сервера не сильно порушить процес побудови відповіді;

Переваги та недоліки:

+ Дозволяє використовувати один і той самий код для створення різноманітних продуктів.

- Клієнт буде прив’язаний до конкретних класів будівельників, тому що в інтерфейсі будівельника може не бути методу отримання результату.

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.

Хід роботи

Реалізація патерну Builder

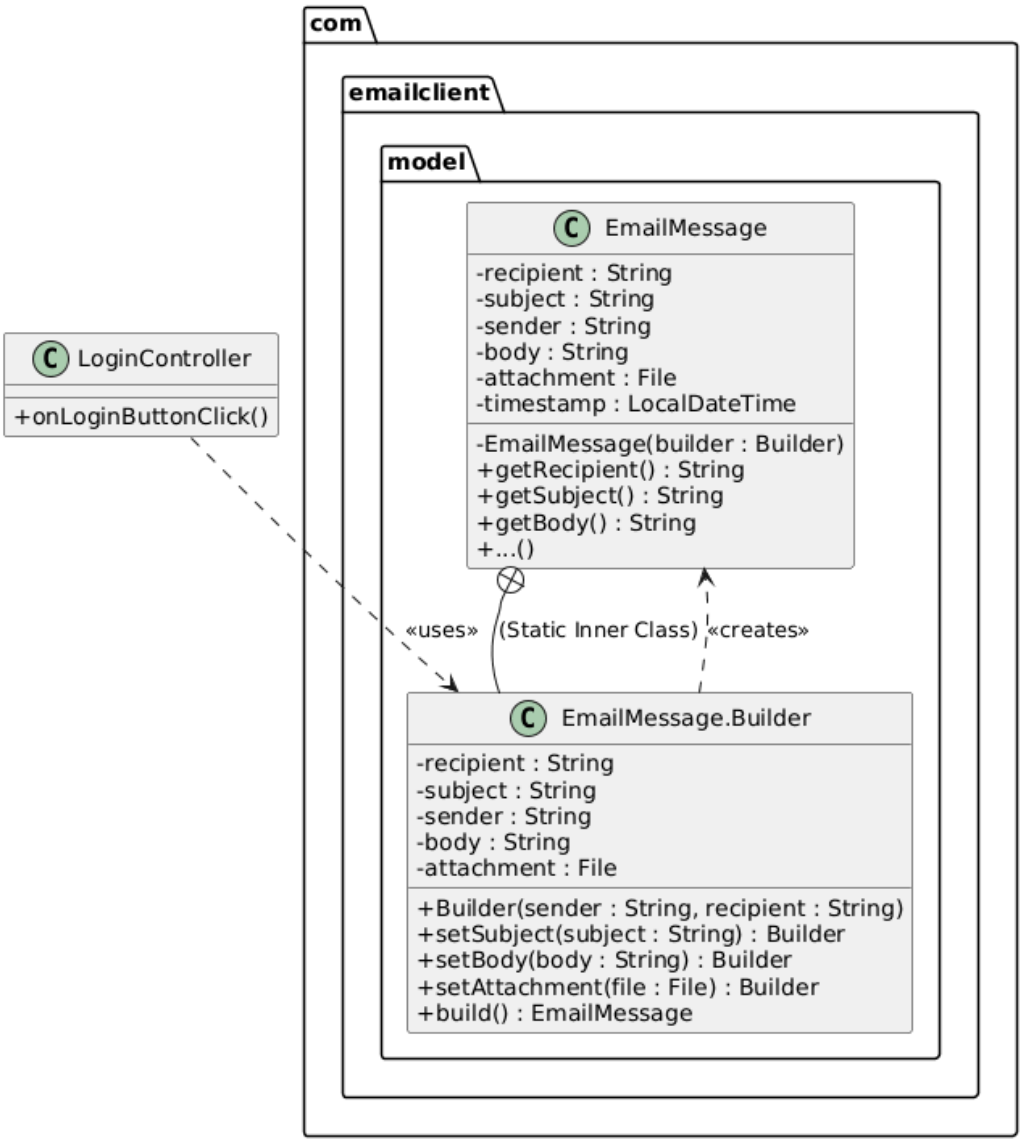

Рис. 1. Діаграма класів з використанням патерну Builder

Фрагмент коду з реалізацією патерну Builder

```
package com.emailclient.model;

import java.io.File;
import java.time.LocalDateTime;

/**
* Патерн: Builder.
* Клас представляє електронний лист.
* Використання Builder дозволяє створювати об'єкт покроково,
* уникаючи конструкторів з великою кількістю параметрів.
*/
public class EmailMessage {
   // Всі поля final, щоб об'єкт був незмінним (Immutable)
   private final String recipient;    // Обов'язкове
   private final String subject;      // Обов'язкове
   private final String sender;       // Обов'язкове

   private final String body;         // Опціональне
   private final File attachment;     // Опціональне
   private final LocalDateTime timestamp; // Автоматичне

   // Приватний конструктор: об'єкт можна створити ТІЛЬКИ через Builder
   private EmailMessage(Builder builder) {
       this.recipient = builder.recipient;
       this.subject = builder.subject;
       this.sender = builder.sender;
       this.body = builder.body;
       this.attachment = builder.attachment;
       this.timestamp = LocalDateTime.now();
   }

   // Геттери (сетерів немає, бо об'єкт Immutable)
   public String getRecipient() { return recipient; }
   public String getSubject() { return subject; }
   public String getSender() { return sender; }
   public String getBody() { return body; }
   public File getAttachment() { return attachment; }
   public LocalDateTime getTimestamp() { return timestamp; }

   @Override
   public String toString() {
       return "EmailMessage{" +
```

```
                "From='" + sender + '\'' +
                ", To='" + recipient + '\'' +
                ", Subject='" + subject + '\'' +
                ", HasAttachment=" + (attachment != null) +
                '}';
    }

    // --- STATIC INNER CLASS: BUILDER ---
    public static class Builder {
        // Ті самі поля, що і в основному класі
        private String recipient;
        private String subject;
        private String sender;
        private String body = ""; // Значення за замовчуванням
        private File attachment;

        // Конструктор Builder-а приймає обов'язкові поля
        public Builder(String sender, String recipient) {
            this.sender = sender;
            this.recipient = recipient;
        }

        // Методи для встановлення опціональних параметрів
        // Вони повертають 'this', щоб можна було писати ланцюжком (Fluent Interface)

        public Builder setSubject(String subject) {
            this.subject = subject;
            return this;
        }

        public Builder setBody(String body) {
            this.body = body;
            return this;
        }

        public Builder setAttachment(File attachment) {
            this.attachment = attachment;
            return this;
        }

        // Метод побудови фінального об'єкта
        public EmailMessage build() {
            if (subject == null) {
```

```
            this.subject = "(Без теми)"; // Логіка валідації або дефолтних значень
        }
        return new EmailMessage(this);
    }
  }
}
```

Висновок: я вивчив структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчився застосовувати їх в реалізації програмної системи.