МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

# Звіт

з лабораторної роботи № 4

з дисципліни «Ефективність та якість архітектурних рішень інформаційних систем»

Виконав:

студент групи ІКМ-М225В

Суліма Нікіта Володимирович

Перевірив:

аспірант каф. КМПС Хорошун Андрій Сергійович

Харків 2025

Інтерфейс Notification

Задає єдиний формат повідомлень для всіх типів сповіщень.

```
class Notification {
public:
    virtual void send(string title, string message) = 0;
    virtual ~Notification() {}
};
```

Існуючий клас EmailNotification

Реалізація відправки звичайних e-mail повідомлень.

```
class EmailNotification : public Notification {
private:
    string adminEmail;

public:
    EmailNotification(string email) : adminEmail(email) {}

    void send(string title, string message) override {
        cout << "[Email] Надіслано лист на " << adminEmail
             << "  Тема: " << title
             << "  Повідомлення: " << message << endl;
    }
};
```

Класи сторонніх сервісів

- **SlackService** — для надсилання повідомлень у Slack.
- **SmsService** — для надсилання SMS.

```
class SlackService {
private:
    string login, apiKey, chatId;
public:
    SlackService(string login, string apiKey, string chatId)
        : login(login), apiKey(apiKey), chatId(chatId) {}
    void auth() { ... }
    void sendToChat(string message) { ... }
};

class SmsService {
private:
    string phone, sender;
public:
    SmsService(string phone, string sender)
        : phone(phone), sender(sender) {}
```

```
    void connect() { ... }
    void sendSMS(string text) { ... }
};
```

## Адаптери

**SlackNotificationAdapter** — адаптує SlackService до Notification:

```
class SlackNotificationAdapter : public Notification {
private:
    SlackService* slack;
public:
    SlackNotificationAdapter(SlackService* service) : slack(service) {}
    void send(string title, string message) override {
        slack->auth();
        slack->sendToChat(title + ": " + message);
    }
};
```

**SmsNotificationAdapter** — адаптує SmsService до Notification:

```
class SmsNotificationAdapter : public Notification {
private:
    SmsService* sms;
public:
    SmsNotificationAdapter(SmsService* service) : sms(service) {}
    void send(string title, string message) override {
        sms->connect();
        sms->sendSMS(title + " — " + message);
    }
};
```

## Клієнтський код

```
int main() {
    SetConsoleOutputCP(65001);

    Notification* email = new EmailNotification("admin@mail.com");
    email->send("Вітання", "Ваш лист доставлено успішно!");

    cout << "---------------------------" << endl;

    SlackService* slackService = new SlackService("user123", "ABC-KEY-999", "dev-chat");
    Notification* slack = new SlackNotificationAdapter(slackService);
    slack->send("Нове повідомлення", "Код виконано без помилок.");

    cout << "---------------------------" << endl;

    SmsService* smsService = new SmsService("+380123456789", "System");
    Notification* sms = new SmsNotificationAdapter(smsService);
    sms->send("Попередження", "Закінчується місце на диску.");
```

```
    delete email;
    delete slack;
    delete sms;
    delete slackService;
    delete smsService;
}
```

Програма створює єдиний інтерфейс Notification, який визначає метод send(). EmailNotification — вже існуюча реалізація для відправки листів. Класи SlackService та SmsService мають власні унікальні методи, тому не відповідають інтерфейсу Notification. Щоб інтегрувати їх у систему, створено адаптери: SlackNotificationAdapter — викликає auth() і sendToChat(), SmsNotificationAdapter — викликає connect() і sendSMS(). Таким чином, клієнтський код може працювати з будь-яким типом повідомлення через один інтерфейс Notification.

## Висновок

У ході лабораторної роботи було ознайомлено з патерном “Адаптер”, створено універсальний інтерфейс для різних систем сповіщень, реалізовано адаптери для Slack та SMS, не змінюючи існуючий код. Патерн “Адаптер” забезпечив гнучкість та розширюваність системи. Клієнтський код не залежить від конкретних реалізацій сервісів — нові типи сповіщень можна легко додавати через нові адаптери.