МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Звіт

з лабораторної роботи №8

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

Базовий клас — EntityUpdater

Це **абстрактний клас**, який описує *загальний процес оновлення сутності* через шаблонний метод update().

```
void update(int entityId, map<string, string> newData)
```

Це і є **шаблонний метод** — він визначає кроки алгоритму:

1. getEntity() — отримати об’єкт із бази або API
2. validateData() — перевірити правильність нових даних
3. Якщо дані невалідні → викликати onValidationFailed() (hook)
4. buildSaveRequest() — сформувати запит
5. sendRequest() — відправити запит (імітація REST API виклику)
6. formatResponse() — підготувати фінальну відповідь
7. afterUpdateHook() — завершальні дії після оновлення

Усі ці методи — віртуальні, тобто кожен підклас може перевизначити свою логіку.

Реалізація для ProductUpdater

Описує, як саме оновлюється товар:

- Якщо ціна (price) некоректна (≤0 або порожня) → валідація не пройдена
- Викликається onValidationFailed(), яка “надсилає сповіщення адміністратору”
- Решта кроків — симуляція надсилання запиту

```
    if (newData["price"].empty() || stoi(newData["price"]) <= 0)
        return false;
```

Реалізація для UserUpdater

Описує, як оновлюється користувач:

- Метод validateData() забороняє змінювати email
  → якщо користувач намагається змінити його, поле видаляється
- Інші кроки виконуються звичайно

```
    if (newData.find("email") != newData.end()) {
```

```
    cout << "[User] Поле 'email' заборонено змінювати! Видаляємо
його.\n";
    newData.erase("email");
}
```

Реалізація для OrderUpdater

Описує, як оновлюється замовлення:

- Виконується звичайна валідація (пропускається)
- Відправляється запит
- У відповіді формується розширений JSON з деталями замовлення - це перевизначення методу formatResponse()

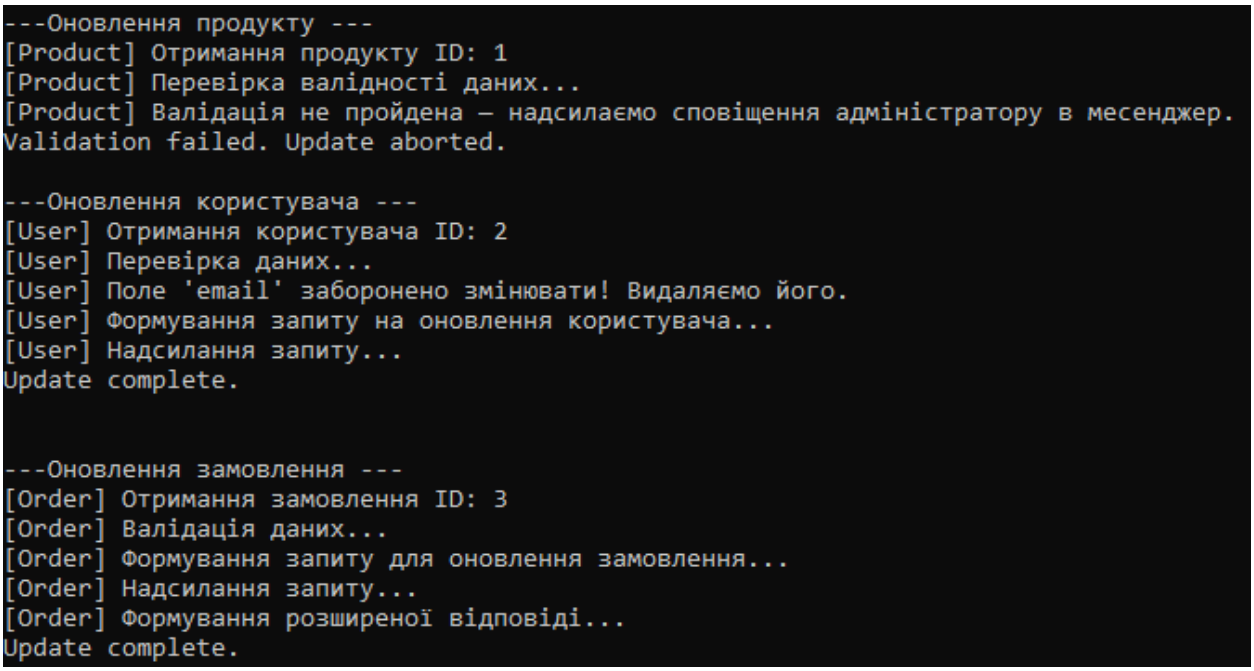
```
string orderInfo = "{id: " + entity["id"] + ", item: " + entity["item"] + ", total: " + entity["total"] +
"}";
return {{"status", "success"}, {"code", "200"}, {"order_data", orderInfo}};
```

Хуки (hooks)

Вони дають можливість “втрутитися” у процес без зміни основного алгоритму:

- onValidationFailed() — викликається при невдалій валідації
- afterUpdateHook() — викликається після оновлення

Результат



Висновок

У ході виконання лабораторної роботи було розглянуто та реалізовано патерн проєктування «Шаблонний метод» на прикладі системи оновлення сутностей — Товар (Product), Користувач (User) та Замовлення (Order). Було створено абстрактний базовий клас EntityUpdater, який визначає загальний алгоритм оновлення даних через метод update(). Цей метод описує послідовність кроків: отримання об’єкта, валідацію даних, формування запиту, відправлення та формування відповіді.