Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

ЗВІТ

про виконання лабораторної роботи №5

Виконав:  
Студент групи ФеП-32с  
Константінов Микита  
  
Перевірив:  
ас. Гура В. Т.

Львів – 2025

1. Мета роботи

Мета лабораторної роботи — провести рефакторинг початкового неструктурованого Python-коду з дотриманням принципів SOLID, реалізувати шаблон проєктування Strategy, забезпечити модульне тестування та перевірку якості коду з використанням SonarQube.

2. Завдання

- Провести рефакторинг файлу main.py з понад 150 рядків.  
- Виокремити сутності (класи Exercise, Workout, TrainingService).  
- Реалізувати шаблон проєктування Strategy.  
- Створити тести за допомогою pytest.  
- Провести аналіз покриття тестами за допомогою pytest-cov.  
- Провести аналіз якості коду за допомогою SonarQube.  
- Оформити документацію та структурувати проєкт.

3. Використані технології

- Мова програмування: Python 3.12  
- Система керування версіями: Git + GitHub  
- Тестування: pytest + pytest-cov  
- Аналіз якості коду: SonarQube + coverage.xml  
- Шаблони проєктування: Strategy  
- Розробницьке середовище: Visual Studio Code (VSCode)

4. Унікальність проєкту

На відміну від попередньої реалізації лабораторної роботи, яка була присвячена системі керування рестораном, даний проєкт реалізує унікальну тематику — менеджер тренувань для фітнесу. Проєкт не базується на типовому прикладі, а повністю розроблений самостійно. Було додано власну ієрархію об'єктів, шаблон Strategy для динамічного вибору типу тренування, реалізовано 3 стратегії (Cardio, Strength, Flexibility), інтегровано логіку збереження в JSON, а також повністю протестовано функціонал за допомогою pytest з покриттям понад 83%. Проєкт збережено у структурованому вигляді та інтегровано з GitHub для демонстрації реального життєвого циклу розробки.

5. Дотримання принципів SOLID

У початковій версії коду всі відповідальності були зосереджені в одному файлі — main.py, що призводило до порушень кількох принципів SOLID. У результаті рефакторингу було досягнуто наступного:

* • SRP (Single Responsibility Principle):

Кожен клас тепер виконує лише одну відповідальність: Exercise — описує вправу, Workout — одне тренування, TrainingService — менеджмент тренувань, окремі стратегії — логіку підрахунку калорій.

* • OCP (Open/Closed Principle):

Для додавання нового типу тренування достатньо створити нову стратегію без зміни існуючого коду Workout або Service.

* • LSP (Liskov Substitution Principle):

Усі реалізації TrainingStrategy можуть заміняти базовий клас без порушення логіки (CardioStrategy, StrengthStrategy).

* • ISP (Interface Segregation Principle):

Інтерфейс TrainingStrategy містить лише один метод calculate\_calories — нічого зайвого для реалізацій.

* • DIP (Dependency Inversion Principle):

Workout не залежить від конкретних стратегій, а лише від абстракції TrainingStrategy, яку передає користувач.

6. Критерії оцінювання

Оцінювання здійснюється за такими критеріями:

|  |  |
| --- | --- |
| Критерій | Оцінка (%) |
| Дотримання SOLID | 20% |
| Архітектура / патерни | 20% |
| Тестування (pytest) | 20% |
| Коректність реалізації (логіка) | 20% |
| Оформлення та структура коду | 20% |

Додатково враховано іншу шкалу оцінювання згідно з методичними рекомендаціями:

|  |  |
| --- | --- |
| Критерій | Оцінка (%) |
| Дотримання технічного завдання | 30% |
| Правильність виконання | 20% |
| Якість рефакторингу / патернів | 20% |
| Модульні тести (кількість, якість, покриття) | 30% |