Лабораторна робота №5
Розробка власної лабораторної роботи для курсу
Термін здачі: 23 травня 2025 року

1. Обрана тема лабораторної роботи
Рефакторинг коду з використанням принципів SOLID

2. Мета лабораторної роботи
Рефакторинг вихідного коду з метою усунення порушень принципів SOLID та забезпечення стабільності програми за допомогою модульного тестування.

3. Завдання на виконання лабораторної роботи
Опис проблеми:
Вихідний код порушує низку принципів SOLID: має зайву відповідальність, тісне зчеплення між класами, важко модифікується та тестується. Це призводить до складнощів у підтримці та розширенні програми.

Технічні вимоги:
- Мова реалізації: Python 3.11
- Фреймворк тестування: pytest
- Мінімум 10 модульних тестів
- Вихідний код: частина завдання
- Заборонено змінювати загальну логіку роботи — лише рефакторинг для відповідності SOLID

Очікувані результати:
- Чистий, структурований, рефакторизований код
- Тести повинні пройти успішно
- Програма повинна зберігати вихідну функціональність
- Покриття тестами основних гілок логіки

4. Технічне завдання
Мова програмування: Python 3.11
Фреймворк тестування: pytest (версія 7.4.0 або новіша)
Структура проєкту:
lab5_solid_refactor/
├── original_code.py
├── refactored_code.py
├── interfaces.py
└── tests/
    └── test_refactored.py
Залежності: лише стандартна бібліотека Python + pytest

5. Приклад коду
Код додається у файлах: original_code.py, refactored_code.py, tests/test_refactored.py

6. Критерії оцінювання
30% — Дотримання технічного завдання
20% — Правильність виконання
20% — Якість рефакторингу згідно з SOLID
30% — Модульні тести (кількість, якість, покриття)

7. Підготовка звіту
Назва:
Рефакторинг коду з використанням принципів SOLID

Мета:
Рефакторинг вихідного коду з використанням принципів SOLID та забезпечення стабільності програми за допомогою модульних тестів.

Коротке пояснення обраної теми:
SOLID — це набір з 5 принципів об’єктно-орієнтованого програмування, які підвищують гнучкість, масштабованість і підтримуваність коду. У даній лабораторній було виявлено порушення SRP (Single Responsibility), а також DIP (Dependency Inversion), які були усунені під час рефакторингу.

Завдання:
- Проаналізувати вихідний код
- Виявити порушення SOLID
- Провести рефакторинг коду
- Написати мінімум 10 модульних тестів
- Скласти звіт

Код рішення: Доданий у файлах refactored_code.py, interfaces.py, tests/

Результати тестування:
Всі 10 тестів проходять успішно. Покриття 100% основної логіки.

Висновки:
Рефакторинг дозволив зробити код більш підтримуваним, кожен клас має одну відповідальність, логіка розділена на окремі компоненти. Структура стала модульною, легко масштабованою. Здобуто навички застосування SOLID на практиці.