

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж



Лабораторна робота №9
з дисципліни Спеціалізовані мови програмування
на тему
Створення та рефакторинг програмно-інформаційного продукту засобами
Python

Виконав:
студент групи РІ-21сп
Владислав РИБАК

Львів – 2024

Мета виконання лабораторної роботи: розробка програмно-інформаційного продукту засобами Python.

План роботи

Завдання 1. Створити скрипт запуску лабораторних робіт 1-8 (Runner) з єдиним меню для управління додатками використовуючи патерн FACADE <https://refactoring.guru/uk/design-patterns/facade>

Завдання 2. Зробити рефакторинг додатків, які були зроблені в лб 1-8, для підтримки можливості запуску через Runner

Завдання 3. Зробити рефакторинг додатків, які були зроблені в лб 1-8, використовуючи багаторівневу архітектуру додатків (див. приклад нижче) та всі принципи об'єктно-орієнтованого підходу

Завдання 4. Створити бібліотеку класів, які повторно використовуються у всіх лабораторних роботах та зробити рефакторинг додатків для підтримки цієї бібліотеки. Таких класів в бібліотеці має бути як найменш 5

Завдання 5. Додати логування функцій в класи бібліотеки програмного продукту використовуючи <https://docs.python.org/uk/3/howto/logging.html>

Завдання 6. Додати коментарі до програмного коду та сформувати документацію програмного продукту засобами roudoc. Документація має бути представлена у вигляді сторінок тексту на консолі, подана у веб-браузері та збережена у файлах HTML

Завдання 7. Документація та код програмного продукту має бути розміщено в GIT репо

Завдання 8. Проведіть статичний аналіз коду продукту засобами PYLINT <https://pylint.readthedocs.io/en/stable/> та виправте помилки, які були ідентифіковані. Первинний репорт з помилками додайте до звіту лабораторної роботи

Завдання 9. Підготуйте звіт до лабораторної роботи

Текст програмної реалізації:

runner.py:

```
import sys
```

```
import os
```

```
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
```

```
sys.path.append(os.path.join(os.path.dirname(os.path.abspath(__file__)), "lab_3"))
```

```
sys.path.append(os.path.join(os.path.dirname(os.path.abspath(__file__)), "lab_4"))
```

```
sys.path.append(os.path.join(os.path.dirname(os.path.abspath(__file__)), "lab_5"))
```

```
sys.path.append(os.path.join(os.path.dirname(os.path.abspath(__file__)), "lab_7"))
```

```
sys.path.append(os.path.join(os.path.dirname(os.path.abspath(__file__)), "lab_8"))
```

```
lab_3_venv_path = os.path.join(
```

```
    os.path.dirname(os.path.abspath(__file__)), "lab_3", "venv", "lib", "python3.13",  
    "site-packages"
```

```

)

sys.path.append(lab_3_venv_path)


lab_5_venv_path = os.path.join(
    os.path.dirname(os.path.abspath(__file__)), "lab_5", "venv", "lib", "python3.13",
    "site-packages"
)

sys.path.append(lab_5_venv_path)


lab_7_venv_path = os.path.join(
    os.path.dirname(os.path.abspath(__file__)), "lab_7", "venv", "lib", "python3.13",
    "site-packages"
)

sys.path.append(lab_7_venv_path)


lab_8_venv_path = os.path.join(
    os.path.dirname(os.path.abspath(__file__)), "lab_8", "venv", "lib", "python3.13",
    "site-packages"
)

sys.path.append(lab_8_venv_path)


from lab_1.main import Command as Lab1Command
from lab_2.main import Command as Lab2Command
from lab_3.main import Command as Lab3Command
from lab_4.main import Command as Lab4Command
from lab_5.main import Command as Lab5Command

```

```
from lab_6.main import Command as Lab6Command
from lab_7.main import Command as Lab7Command
from lab_8.main import Command as Lab8Command
```

```
class Runner:
```

```
    def __init__(self):
```

```
        self.commands = {}
```

```
        self._initialize_commands()
```

```
    def _initialize_commands(self):
```

```
        try:
```

```
            self.commands["1"] = Lab1Command()
```

```
            self.commands["2"] = Lab2Command()
```

```
            self.commands["3"] = Lab3Command()
```

```
            self.commands["4"] = Lab4Command()
```

```
            self.commands["5"] = Lab5Command()
```

```
            self.commands["6"] = Lab6Command()
```

```
            self.commands["7"] = Lab7Command()
```

```
            self.commands["8"] = Lab8Command()
```

```
        except ImportError as e:
```

```
            print(f"Помилка при завантаженні команд: {e}")
```

```
    def display_menu(self):
```

```
        print("\nМеню запуску лабораторних робіт:")
```

```
        if not self.commands:
```

```

    print("Немає доступних лабораторних робіт")

    return

for number in sorted(self.commands.keys()):

    print(f"Лабораторна робота №{number}")

print("Введіть номер лабораторної роботи або 'q' для виходу")


def start(self):

    print("Ласкаво просимо до системи виконання лабораторних робіт!")


while True:

    try:

        self.display_menu()

        choice = input("Ваш вибір: ").strip().lower()


        if choice == 'q':

            print("Дякуємо за використання програми!")

            break


        if not choice:

            print("Будь ласка, зробіть вибір")

            continue


        command = self.commands.get(choice)

        if command:

```

```

        command.execute()

    else:

        print("Невірний номер лабораторної роботи")

except KeyboardInterrupt:

    print("\nПрограму завершено користувачем")

    break

except Exception as e:

    print(f"Виникла помилка: {e}")

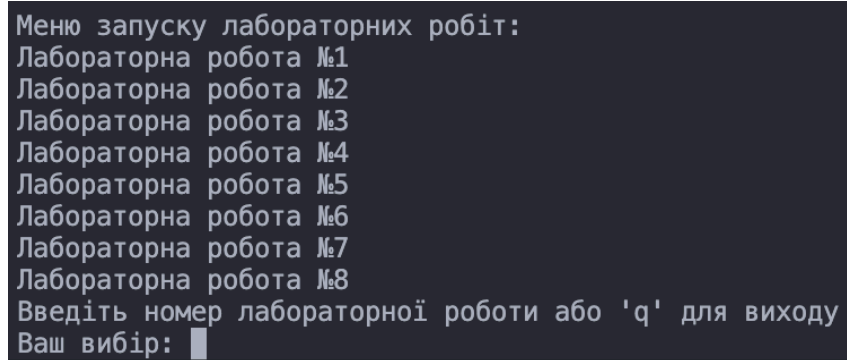
if __name__ == "__main__":

    runner = Runner()

    runner.start()

```

Результати роботи:



```

Меню запуску лабораторних робіт:
Лабораторна робота №1
Лабораторна робота №2
Лабораторна робота №3
Лабораторна робота №4
Лабораторна робота №5
Лабораторна робота №6
Лабораторна робота №7
Лабораторна робота №8
Введіть номер лабораторної роботи або 'q' для виходу
Ваш вибір: █

```

Рис. 1. Консольний інтерфейс

```

Меню запуску лабораторних робіт:
Лабораторна робота №1
Лабораторна робота №2
Лабораторна робота №3
Лабораторна робота №4
Лабораторна робота №5
Лабораторна робота №6
Лабораторна робота №7
Лабораторна робота №8
Введіть номер лабораторної роботи або 'q' для виходу
Ваш вибір: 3

--- ASCII Art Generator ---
1. Створити новий ASCII-арт
2. Вийти
Виберіть опцію (1-2): █

```

Рис. 2. Результат запуску 3 лабораторної

```

Ваш вибір: 7
1. Отримати та відобразити дані
2. Зберегти дані у файл
3. Вийти
Оберіть опцію: 1
Введіть API-ендпоінт (наприклад, posts): posts

```

userId	id	title	body
1	1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit
			suscipit recusandae consequuntur expedita et cum

Рис. 3. Результат запуску 7 лабораторної

Висновки: на цій лабораторній роботі я розробив програмно-інформаційний продукт засобами Python.