МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж

Лабораторна робота №9

з дисципліни: «Спеціалізовані мови програмування»

на тему: «Створення та рефакторінг програмно-інформаційного продукту засобами Python»

Виконала:

студентка групи ІТ-32

Моляща Ю.А.

Прийняв:

Щербак С.С.

Львів - 2023

Лабораторна робота №9

«Створення та рефакторінг програмно-інформаційного продукту засобами Python»

Мета роботи: розробка програмно-інформаційного продукту засобами Python.

Завдання на лабораторну роботу.

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 рефакторінг 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 принципи об'єктно-орієнтованого підходу

Завдання 4. Створити бібліотеку класів, які повторно використовуються у всіх лабораторних роботах та зробити рефакторінг додатків для підтримки цієї бібліотеки. Таких класів в бібліотеці має буде 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 pydoc. 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 repo

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 лабораторной роботи

Хід роботи.

Завдання 1.

Створити скрипт запуску лабораторних робіт 1-8 (Runner) з єдиним меню для управління додатками використовуючи патерн FACADE

Код програми:

```
import importlib


    class Runner:
```

```
            """
            This class provides an interface for users to select and execute labs
            within the specified range (Lab 1 to Lab 8). It utilizes a modular
            approach, importing the 'main' function from the respective lab
modules.
            """
            MAX_LAB_NUMBER = 8

            @staticmethod
            def run_lab(lab_number):
                """Run the selected lab."""
                while True:
                    try:
                        # Check if lab is a test suite
                        if lab_number == 6:
                            print(f"Lab {lab_number} is a test suite. Please run
the test suite instead.")
                            break

                        # Import the 'main' function from the lab module
                        module_name = f"ui.lab{lab_number}.lab{lab_number}_menu"
                        main_module = importlib.import_module(module_name)
                        main_function = getattr(main_module, "main")

                        # Check if 'main' is callable
                        if callable(main_function):
                            main_function()
                            print(f"Lab {lab_number} executed successfully.")
                            break
                        else:
                            print(f"Error: 'main' function not found in module
{module_name}")
                            break

                    except ImportError as e:
```

```
                    print(f"Error importing module {module_name}: {e}")
                    break

        def get_user_input(self):
            """Get user input for lab selection."""
            while True:
                try:
                    print(f"\nSelect a lab to run (1-{self.MAX_LAB_NUMBER})
or press 'q' to quit:")
                    user_input = input("Lab: ")

                    if user_input.lower() == 'q':
                        print("Exiting...")
                        return None

                    lab_number = int(user_input)
                    if 1 <= lab_number <= self.MAX_LAB_NUMBER:
                        return lab_number
                    else:
                        print(f"Please enter a number between 1 and
{self.MAX_LAB_NUMBER}.")

                except ValueError:
                    print("Please enter a valid integer.")

        def main(self):
            while True:
                lab_number = self.get_user_input()
                if lab_number is None:
                    break
                self.run_lab(lab_number)

    if __name__ == "__main__":
        Runner().main()
```

Завдання 2.

Виконання завдання 2 зображено на рис. 1.

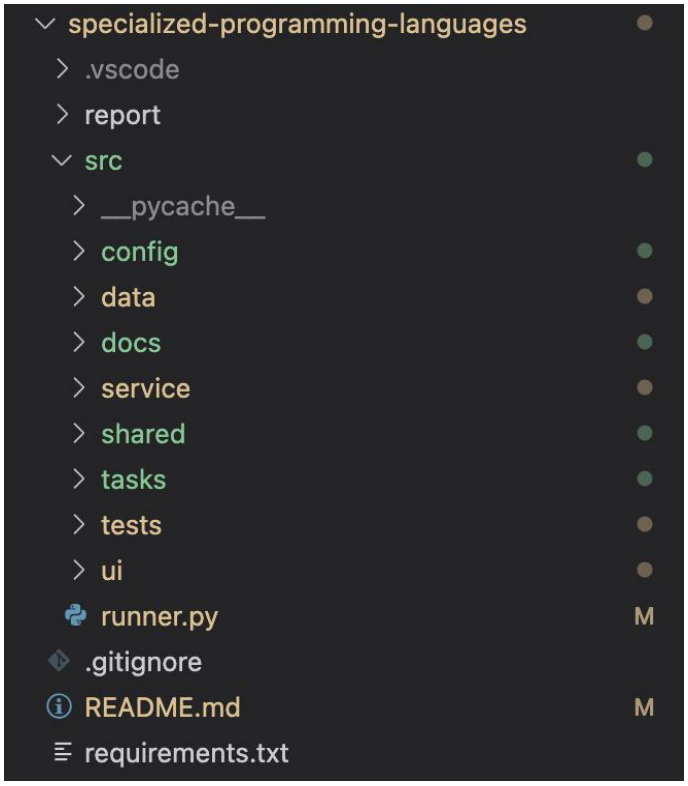

Рис. 1 Рефакторинг використовуючи багаторівневу архітектуру додатків

Завдання 3.

Виконання завдання 3 зображено на рис. 1.

Завдання 4.

Виконання завдання 4 зображено на рис. 2.

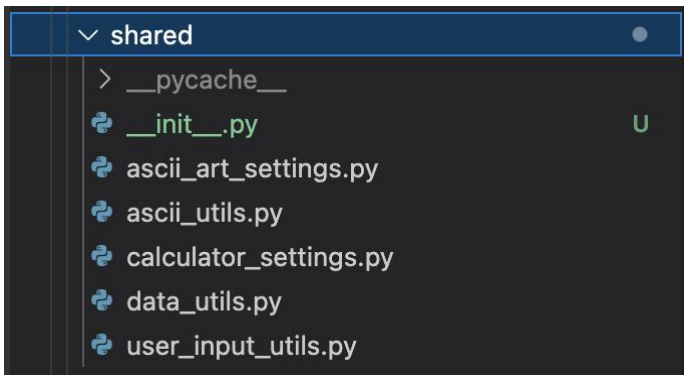

Рис. 2 Бібліотека класів

Завдання 5.

Використання логування на прикладі класу AsciiArtSettings.

Код програми:

```
"""Module for storing settings for ASCII-art."""
import json
import logging

logging.basicConfig(level=logging.INFO)

class AsciiArtSettings:
    """Class for storing settings for ASCII-art.

    Attributes:
        font (str): Font name.
        size (tuple): Size of ASCII-art.
        symbols (str): Symbols to replace.
        color (str): Color of ASCII-art.
    """
    def __init__(self):
        self.settings_file_path = None
        self.font = None
        self.size = None
        self.symbols = None
        self.color = None
        self.alignment = None
        self.is_3d = None
        self.logger = logging.getLogger(__name__)

    def set_settings_file_path(self, settings_file_path):
        self.settings_file_path = settings_file_path

    def show_settings(self):
```

```
        print('Current settings:')
        print('Font:', self.font)
        print('Size:', self.size)
        print('Symbols:', self.symbols)
        print('Alignment:', self.alignment)
        print('Color:', self.color.replace('\x1b', '\\x1b'))
        print('3D:', self.is_3d)

    def set_font(self, font):
        self.font = font
        self.save_settings()
        self.logger.info(f'Font changed to {font}')

    def set_size(self, width, height):
        self.size = (width, height)
        self.save_settings()
        self.logger.info(f'Size changed to {width}x{height}')

    def set_symbols(self, regular_symbol, shadow_symbol):
        self.symbols = (regular_symbol, shadow_symbol)
        self.save_settings()
        self.logger.info(f'Symbols changed to {regular_symbol} and
{shadow_symbol}')

    def set_color(self, color):
        self.color = color
        self.save_settings()
        self.logger.info(f'Color changed to {color}')

    def set_alignment(self, alignment):
        self.alignment = alignment
        self.save_settings()
        self.logger.info(f'Alignment changed to {alignment}')
```

```
def set_3d_option(self, is_3d):
    self.is_3d = is_3d
    self.save_settings()
    self.logger.info(f'3D option changed to {is_3d}')

def default_settings(self):
    self.font = 'clb6x10'
    self.size = (80, 25)
    self.symbols = ("#", "*")
    self.color = '\x1b[39m'
    self.alignment = 'left'
    self.is_3d = False
    self.save_settings()
    self.logger.info('Default settings loaded')

def save_settings(self):
    settings_data = {
        'font': self.font,
        'size': self.size,
        'symbols': self.symbols,
        'color': self.color,
        'alignment': self.alignment,
        'is_3d': self.is_3d
    }
    with open(self.settings_file_path, 'w') as file:
        json.dump(settings_data, file)
    self.logger.info('Settings saved')

def load_settings(self):
    try:
        with open(self.settings_file_path, 'r') as file:
            settings_data = json.load(file)
            self.font = settings_data['font']
            self.size = settings_data['size']
```

```
            self.symbols = settings_data['symbols']
            self.color = settings_data['color']
            self.alignment = settings_data['alignment']
            self.is_3d = settings_data['is_3d']
    except FileNotFoundError:
        # If file does not exist use default settings
        self.default_settings()
    self.logger.info('Settings loaded')
```

Завдання 6.

Сформуємо документацію. Результат зображено на Рис.3.

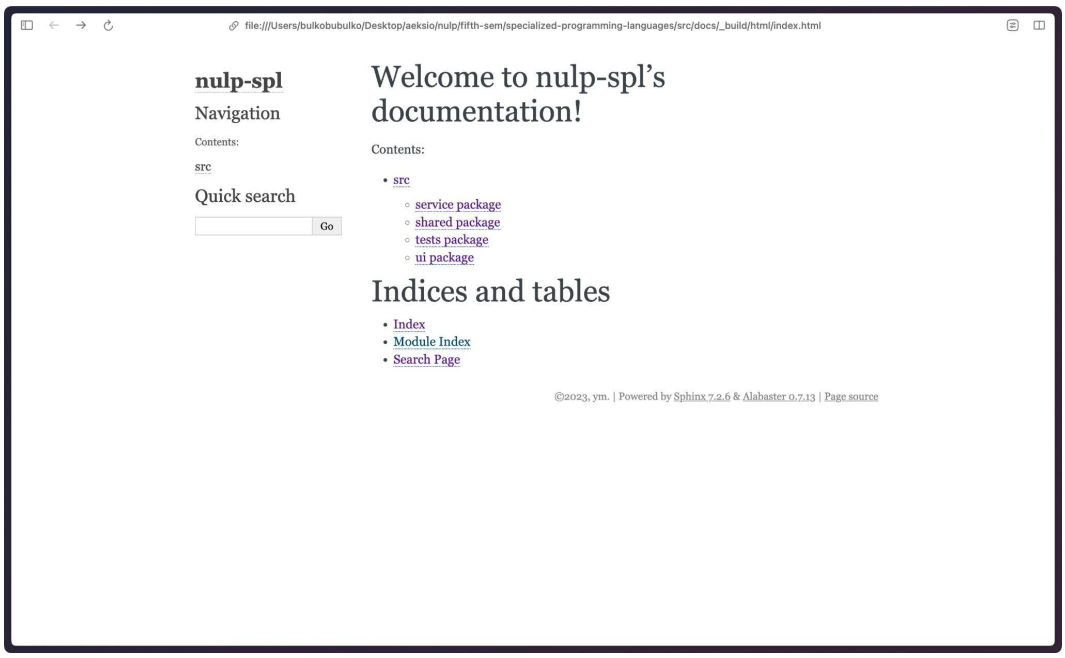

Рис. 3 Згенерована документація

Завдання 7.

Посилання на GitHun-репозиторій: https://github.com/bulkobubulko/nulp-spl

Завдання 8.

```
--------------------------------------------------------------------
Your code has been rated at 5.10/10 (previous run: 5.10/10, +0.00)
```

Рис. 4 Статистичний аналіз коду засобом PYLINT

Висновок: під час виконання лабораторної роботи було розроблено програмно-інформаційного продукту засобами Python.