МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №9

з курсу

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Виконав студент

групи ІТ-21сп

**Гузар А.І.**

Прийняв

**Щербак С.С.**

Львів - 2023

**Мета 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 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

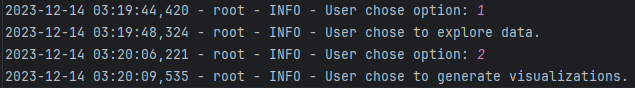
**Завдання 8.**

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

**Код програми:**  
class Run:  
 """  
 The Run class runs different labs.  
 """  
  
 def \_\_init\_\_(self):  
 """  
 Initializes a Run object with a reader/writer and a validator.  
 """  
 self.reader\_writer = ReaderWriter()  
 input\_parser = InputParser()  
 self.validator = Validator(input\_parser)  
  
 def run\_lab1(self):  
 """  
 Runs lab 1.  
 """  
 calculator = BasicCalculator()  
 menu = MenuLab1(calculator, self.validator)  
 menu.run()  
  
 def run\_lab2(self):  
 """  
 Runs lab 2.  
 """  
 calculator = Calculator()  
 menu = MenuLab2(calculator)  
 menu.run()  
  
 def run\_lab3(self):  
 """  
 Runs lab 3.  
 """  
 generator = ASCIIArtGenerator(self.reader\_writer, self.validator)  
 menu = MenuLab3(generator, self.validator, self.reader\_writer)  
 menu.run()  
  
 def run\_lab4(self):  
 """  
 Runs lab 4.  
 """  
 generator = ASCIIArtGenerate(ascii\_art, self.reader\_writer)  
 menu = MenuLab4(generator, self.reader\_writer, self.validator)  
 menu.run()  
  
 def run\_lab5(self):  
 """  
 Runs lab 5.  
 """  
 menu = MenuLab5(self.reader\_writer, self.validator)  
 menu.run()  
  
 def run\_lab6(self):  
 """  
 Runs lab 6.  
 """  
 suite = unittest.TestLoader().loadTestsFromTestCase(UnitTestsCalculator)  
 unittest.TextTestRunner().run(suite)  
  
 def run\_lab7(self):  
 """  
 Runs lab 7.  
 """  
 api\_client = APIClient(api\_url='https://jsonplaceholder.typicode.com/todos/1')  
 input\_parser = InputParser()  
 history = History()  
 error\_handler = ErrorHandler()  
 results\_display = ResultsDisplay()  
 data\_saver = DataSaver()  
  
 api\_app = APIApp(api\_client, self.reader\_writer, self.validator, history, error\_handler, results\_display, input\_parser, data\_saver)  
 menu\_lab7 = MenuLab7(api\_app, self.validator, self.reader\_writer)  
 menu\_lab7.run()  
  
 def run\_lab8(self):  
 """  
 Runs lab 8.  
 """  
 data\_processor = DataProcessor()  
 menu = MenuLab8(data\_processor, self.validator, self.reader\_writer)  
 menu.run()  
  
 def run(self):  
 """  
 Runs the specified lab or quits.  
 """  
 while True:  
 lab = input("Enter the number of the lab (1-8) or 'q' to quit: ")  
 if lab == '1':  
 self.run\_lab1()  
 elif lab == '2':  
 self.run\_lab2()  
 elif lab == '3':  
 self.run\_lab3()  
 elif lab == '4':  
 self.run\_lab4()  
 elif lab == '5':  
 self.run\_lab5()  
 elif lab == '6':  
 self.run\_lab6()  
 elif lab == '7':  
 self.run\_lab7()  
 elif lab == '8':  
 self.run\_lab8()  
 elif lab.lower() == 'q':  
 print("Goodbye!")  
 break  
 else:  
 print(f"No results: {lab}. Please try again.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 Run().run()

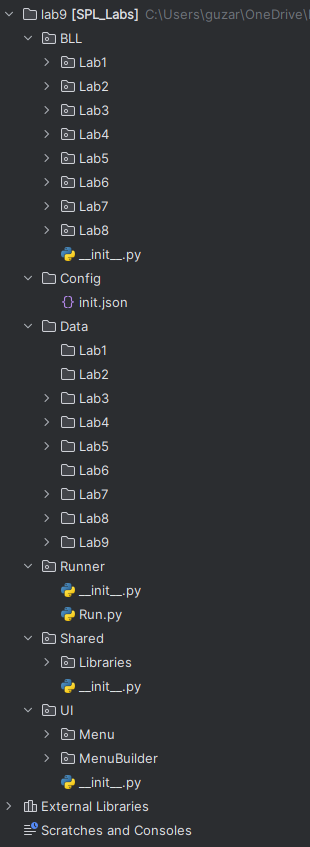
class MenuBuilder:  
 """  
 A class to build and display a menu.  
 """  
  
 def \_\_init\_\_(self, config):  
 """  
 Initializes the MenuBuilder with a configuration.  
  
 Args:  
 config (dict): The configuration for the menu.  
 """  
 self.version = config['version']  
 self.title = config['title']  
 self.options = config['options']  
 self.is\_vertical = config['is\_vertical']  
  
 def display(self):  
 """  
 Displays the menu to the user.  
 """  
 max\_option\_length = max([len(option) for option in self.options])  
 border\_length = max\_option\_length \* len(self.options) + len(  
 self.options) \* 5 - 1 if not self.is\_vertical else len(self.title) + 2  
 border = "+" + "-" \* border\_length + "+"  
 title\_padding = (border\_length - len(self.title)) // 2  
 print(border)  
 print("|" + " " \* title\_padding + "\033[1;31m" + self.title + "\033[m" + " " \* (  
 border\_length - title\_padding - len(self.title)) + "|") # Red color  
 print(border)  
 for i, option in enumerate(self.options):  
 print("\033[1;32m" + (str(i + 1) + ". " if self.is\_vertical else str(i + 1) + ". ") + option + "\033[m",  
 end=("\n" if self.is\_vertical else " ")) # Green color  
 print(border)  
 print("\033[1;34m" + f"Menu Version: {self.version}" + "\033[m") # Blue color

На рисунку 1 зображено результат виконання логування.



**Рис. 1** Результат виконання логування

На рисунку 2 зображено загальний вигляд проекту.



**Рис. 2** Архітектура проекту

**Висновок:** я створив програмно-інформаційний продукт засобами Python.