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 рефакторінг програмно-інформаційного продукту засобами Python

Виконав:

ст. гр. РІ-21сп

Павлусь В.П

Прийняв:

Шербак С.С.

Львів-2024

**Мета лабораторної роботи:** Розробка програмно-інформаційного продукту засобами Python

**Хід 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 всі 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.

**Виконання роботи**

**Вміст файлу fascade.py:**

from Lab1.runner import main as lab1

from Lab2.interface import main as lab2

from Lab3.UI.user\_interface import main as lab3

from Lab4.UI.user\_interface import main as lab4

from Lab5.UI.user\_interface import user\_interface as lab5

from Lab6.run\_tests\_with\_coverage import main as lab6

from Lab7.UI.user\_interface import main as lab7

from Lab8.UI.user\_interface import main as lab8

class RunnerFacade:

def \_\_init\_\_(self):

self.labs = {

1: self.run\_lab1,

2: self.run\_lab2,

3: self.run\_lab3,

4: self.run\_lab4,

5: self.run\_lab5,

6: self.run\_lab6,

7: self.run\_lab7,

8: self.run\_lab8,

}

def show\_menu(self):

print("\nAll programs:")

for lab\_number in sorted(self.labs.keys()):

print(f"{lab\_number}: Program {lab\_number}.")

print("0: Exit.")

def run\_lab(self, choice):

if choice in self.labs:

print(f"Starting program {choice}...")

self.labs[choice]()

elif choice == 0:

print("Exit!")

else:

print("Error: Incorrect option! Please, try again!")

def run\_lab1(self):

lab1()

def run\_lab2(self):

lab2()

def run\_lab3(self):

lab3()

def run\_lab4(self):

lab4()

def run\_lab5(self):

lab5()

def run\_lab6(self):

lab6()

def run\_lab7(self):

lab7()

def run\_lab8(self):

lab8()

**Вміст файлу main.py:**

from Lab9.facade import RunnerFacade

def main():

facade = RunnerFacade()

while True:

facade.show\_menu()

try:

choice = int(input("Select program: "))

if choice == 0:

print("Exit from final program!")

break

facade.run\_lab(choice)

except ValueError:

print("Error: Incorrect option! Please, try again!")

**Вміст файлу runner.py:**

import sys

import os

current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

root\_path = os.path.abspath(os.path.join(current\_dir, ".."))

sys.path.append(root\_path)

from main import main

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Папка Shared:**

**Вміст файлу components.py:**

import math

from abc import ABC, abstractmethod

memory = 0

history = []

decimal\_places = 2

class Memory:

def \_\_init\_\_(self):

self.\_\_memory = memory

def set\_memory(self, value):

self.\_\_memory = value

def get\_memory(self):

return self.\_\_memory

def clear\_memory(self):

self.\_\_memory = 0

class History:

def \_\_init\_\_(self):

self.\_\_history = history

def add\_to\_history(self, expression, result):

self.\_\_history.append(f"{expression} = {result}")

def show\_history(self):

if self.\_\_history:

print("History of calculations:")

for entry in self.\_\_history:

print(entry)

else:

print("History is empty.")

class Settings:

def \_\_init\_\_(self):

self.\_\_decimal\_places = decimal\_places

def set\_decimal\_places(self):

places = input("Enter the number of symbols after the decimal point: ")

if places.isalpha():

raise places("Please, enter a positive integer!")

elif int(places) >= 0:

self.\_\_decimal\_places = int(places)

print(f"Number of symbols after decimal point - {self.\_\_decimal\_places}.")

else:

raise ValueError("Please, try again. Number of symbols must not be negated!")

def get\_decimal\_places(self):

return self.\_\_decimal\_places

class BaseCalculator(ABC):

def \_\_init\_\_(self):

self.memory = Memory()

self.history = History()

self.settings = Settings()

def get\_input(self):

try:

num1\_input = input("Enter first number or 'm' for use value from memory: ")

num1 = self.memory.get\_memory() if num1\_input.lower() == 'm' else float(num1\_input)

operator = self.get\_operator()

num2 = None

if operator != '√':

while True:

try:

num2\_input = input("Enter second number or 'm' for use value from memory: ")

num2 = self.memory.get\_memory() if num2\_input.lower() == 'm' else float(num2\_input)

break

except ValueError:

print("Incorrect input! Please, try again!")

return num1, operator, num2

except ValueError:

print("Incorrect input! Please, try again!")

return self.get\_input()

def check\_operator(self, operator):

return operator in ['+', '-', '\*', '/', '^', '√', '%']

def get\_operator(self):

while True:

operator = input("Choose one of the operators (+, -, \*, /, ^, √, %): ")

if self.check\_operator(operator):

return operator

else:

print("Incorrect operator! Please, try again!")

@abstractmethod

def perform\_operators(self, num1, operator, num2):

pass

Результат виконання, а саме виведення скрипта запуску лабораторних робіт 1-8 зображено на рис. 1.

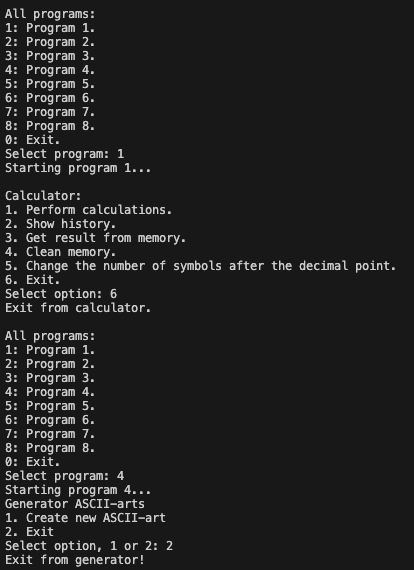


Рис. 1 – Запуск усіх лабораторних робіт

**Висновок:** У ході виконання ЛР я створив скрипт запуску лабораторних робіт 1-8 (Runner) з єдиним меню для управління додатками використовуючи патерн FACADE.