МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж



Лабораторна робота №2

з дисципліни «Спеціалізовані мови програмування»

на тему «Основи побудови об’єктно-орієнтованих додатків на Python»

Виконала студентка

групи РІ-32

Титиш О.І.

Прийняв:

Щербак С.С

Львів – 2024

**Мета:** Розробка консольного калькулятора в об’єктно-орієнтованому стилі з використанням класів.

**План роботи**

Завдання 1: Створення класу Calculator

Створіть клас Calculator, який буде служити основою для додатка калькулятора.

Завдання 2: Ініціалізація калькулятора

Реалізуйте метод \_\_init\_\_ у класі Calculator для ініціалізації необхідних атрибутів або змінних.

Завдання 3: Введення користувача

Перемістіть функціональність введення користувача в метод у межах класу Calculator. Метод повинен приймати введення для двох чисел і оператора.

Завдання 4: Перевірка оператора

Реалізуйте метод у класі Calculator, щоб перевірити, чи введений оператор є дійсним (тобто одним із +, -, \*, /). Відобразіть повідомлення про помилку, якщо він не є дійсним.

Завдання 5: Обчислення

Створіть метод у класі Calculator, який виконує обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення).

Завдання 6: Обробка помилок

Реалізуйте обробку помилок у межах класу Calculator для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідні повідомлення про помилку.

Завдання 7: Повторення обчислень

Додайте метод до класу Calculator, щоб запитати користувача, чи він хоче виконати ще одне обчислення. Якщо так, дозвольте йому ввести нові числа і оператор. Якщо ні, вийдіть з програми.

Завдання 8: Десяткові числа

Модифікуйте клас Calculator для обробки десяткових чисел (плаваюча кома) для більш точних обчислень.

Завдання 9: Додаткові операції

Розширте клас Calculator, щоб підтримувати додаткові операції, такі як піднесення до степеня (^), квадратний корінь (√) та залишок від ділення (%).

Завдання 10: Інтерфейс, зрозумілий для користувача

Покращте інтерфейс користувача у межах класу Calculator, надавши чіткі запити, повідомлення та форматування виводу для зручності читання.

**Код програми**

calculator.py

from DAL.memory\_LAB2 import log\_operation, log\_history  
  
class Calculator:  
 def calculate(self, num1, operator, num2=None):  
 match operator:  
 case '+':  
 result = num1 + num2  
 case '-':  
 result = num1 - num2  
 case '\*':  
 result = num1 \* num2  
 case '/':  
 if num2 == 0:  
 raise ZeroDivisionError("Ділення на нуль!")  
 result = num1 / num2  
 case '^':  
 result = num1 \*\* num2  
 case '%':  
 result = num1 % num2  
 case '√':  
 result = num1 \*\* 0.5  
 case \_:  
 raise ValueError("Невірний оператор")  
   
 *# Формування виразу* expression = f"{num1} {operator} {num2 if operator != '√' else ''}"  
  
 *# Логування операції* log\_operation(f"{expression} = {result}")  
 log\_history(expression, result)  
  
 return result

validator.py

def is\_valid\_operator(operator):  
 return operator in ['+', '-', '\*', '/', '^', '%', '√']

class\_calc.py

from Shared.AppSettings\_LAB1 import decimal\_places  
from Shared.logs.logger\_LAB1 import log\_operation, log\_history, show\_history  
from functions\_LAB1 import calculate  
  
class Calculator:  
 def \_\_init\_\_(self):  
 self.memory = None  
 self.decimal\_places = decimal\_places  
  
 def get\_input(self):  
 try:  
 num1 = float(input("Введіть перше число: "))  
 operator = input("Введіть оператор (+, -, \*, /, ^, %, √): ")  
 num2 = None  
 if operator != '√':  
 num2 = float(input("Введіть друге число: "))  
 return num1, operator, num2  
 except ValueError:  
 print("Неправильний ввід. Спробуйте знову.")  
 return self.get\_input()  
  
 def is\_valid\_operator(self, operator):  
 return operator in ['+', '-', '\*', '/', '^', '%', '√']  
  
 def perform\_calculation(self, num1, operator, num2):  
 try:  
 result = calculate(num1, operator, num2)  
 return round(result, self.decimal\_places)  
 except ZeroDivisionError:  
 print("Помилка: Ділення на нуль неможливе.")  
 except ValueError as e:  
 print(f"Помилка: {e}")  
 except Exception as e:  
 print(f"Виникла непередбачена помилка: {e}")  
 return None  
  
 def ask\_to\_continue(self):  
 return input("Бажаєте виконати ще одне обчислення? (так/ні): ").lower() == 'так'  
  
 def store\_in\_memory(self, result):  
 self.memory = result  
 print(f"Результат {result} збережений у пам'яті.")  
  
 def recall\_memory(self):  
 if self.memory is not None:  
 print(f"Збережене значення: {self.memory}")  
 return self.memory  
 else:  
 print("Пам'ять порожня.")  
 return None  
  
 def run(self):  
 print(f"Результати відображатимуться з {self.decimal\_places} десятковими знаками.")  
 while True:  
 num1, operator, num2 = self.get\_input()  
  
 if not self.is\_valid\_operator(operator):  
 print("Недійсний оператор. Спробуйте ще раз. Ви можете використовувати тільки +, -, \*, /, ^, %, √")  
 continue  
  
 result = self.perform\_calculation(num1, operator, num2)  
 if result is not None:  
 print(f"Результат: {result}")  
 self.store\_in\_memory(result)  
  
 expression = f"{num1} {operator} {num2 if operator != '√' else ''}"  
 log\_operation(f"{expression} = {result}")  
 log\_history(expression, result)  
  
 if input("Бажаєте переглянути історію розрахунків? (так/ні): ").lower() == 'так':  
 show\_history()  
  
 if not self.ask\_to\_continue():  
 break

memory.py

memory = None  
history = []  
  
def store\_in\_memory(result):  
 global memory  
 memory = result  
 print(f"Результат {result} збережений у пам'яті.")  
  
def recall\_memory():  
 if memory is not None:  
 print(f"Збережене значення: {memory}")  
 return memory  
 else:  
 print("Пам'ять порожня.")  
 return None  
  
def log\_operation(operation):  
 *# Збереження операції в історії* history.append(operation)  
  
def log\_history(expression, result):  
 history.append(f"{expression} = {result}")  
  
def show\_history():  
 if history:  
 print("Історія розрахунків:")  
 for entry in history:  
 print(entry)  
 else:  
 print("Історія порожня.")

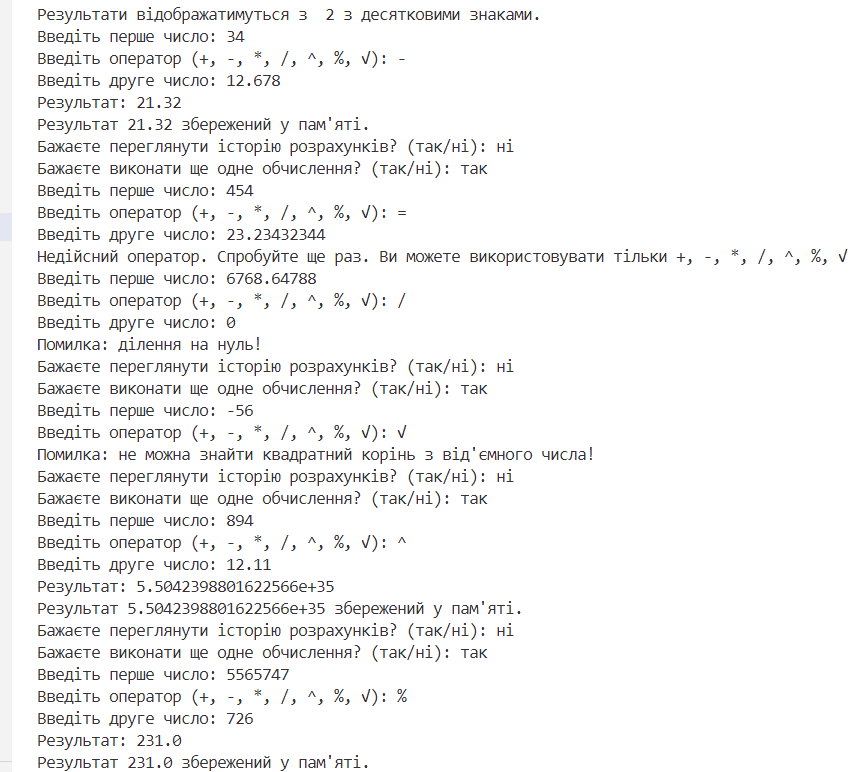
runner.py

from classes.class\_calc\_LAB2 import Calculator  
from BLL.calculator\_LAB2 import Calculator  
from BLL.validator\_LAB2 import is\_valid\_operator  
from DAL.memory\_LAB2 import store\_in\_memory, recall\_memory, show\_history

*# Пам'ять для збереження результатів*memory = None

if \_\_name\_\_ == "\_\_main\_\_":  
 calculator()

**Результат виконання програми**



*Рисунок 1. Результат виконання*

Посилання на GitHub:

**Висновок:** На цій лабораторній роботі було перетворено консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. За допомогою цього проекту я вивчила концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.