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

Виконав:

ст. гр. РІ-31

ЛАЗАР В.С.

Прийняв:

ЩЕРБАК С.С.

Львів-2024

**Мета роботи:**

Розробка консольного калькулятора в об’єктно орієнтованому стилі з використанням класів.

**Хід роботи:**

**Завдання 1: Створення класу Calculator**

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

**Завдання 2: Ініціалізація калькулятора**

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

**Завдання 3: Введення користувача**

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

**Завдання 4: Перевірка оператора**

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

**Завдання 5: Обчислення**

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

**Завдання 6: Обробка помилок**

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

**Завдання 7: Повторення обчислень**

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

**Завдання 8: Десяткові числа**

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

**Завдання 9: Додаткові операції**

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

**Завдання 10: Інтерфейс, зрозумілий для користувача**

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

**Код програми:**

from BLL.classes.calculator import Calculator

from DAL.classes.history import History

from BLL.classes.validators import Validators

from GlobalVariables import memory\_operations

import GlobalVariables as GlobalVariables

class Console:

@staticmethod

def prompt():

case = input("\n1 - Calculate a number \n"

"2 - View history \n"

"3 - Additional settings \n"

"Your choice: ")

match case:

case "1":

return Console.calculator()

case "2":

History.read()

return False

case "3":

Console.settings()

return False

case \_:

return True

@staticmethod

def calculator():

num1 = Validators.validate\_num("\nEnter first number (or MR / MC): ")

operator = Validators.validate\_operator()

if operator in memory\_operations:

Validators.validate\_memory(operator, num1)

return False

num2 = Validators.validate\_num("Enter second number (or MR / MC): ")

if operator == "/" and num2 == 0:

print("Error: cannot divide by zero")

return False

result = Calculator(num1, num2, operator, GlobalVariables.digits)

print("Result : " + str(result.result))

try\_again = input("\nCalculation has finished successfully! \n"

"Current options: \n"

"Try again? (Y / N) \n"

"Store a value into memory? (MS / M+ / M-) \n"

"Your choice: ").lower()

if try\_again in memory\_operations:

Validators.validate\_memory(try\_again, result.result)

elif try\_again == "y":

return False

else:

return True

@staticmethod

def settings():

settings\_prompt = input("\n1 - Change the amount of digits after a decimal point in a number \n"

"2 - Clear history\n"

"Your choice: ")

match settings\_prompt:

case "1":

while True:

digits\_prompt = input("\nEnter the amount of digits (Current value: " + str(GlobalVariables.digits) + "): ")

digits = Validators.validate\_digits(digits\_prompt)

if digits:

print("Settings changed successfully\n")

break

else:

print("Invalid input, please enter a valid non-negative integer number")

case "2":

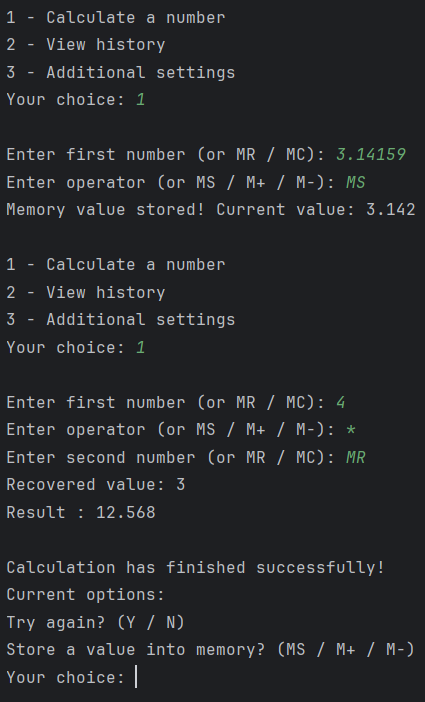
History.clear()

print("History cleared successfully")

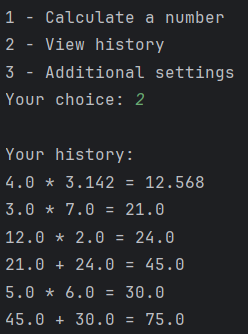
case \_:

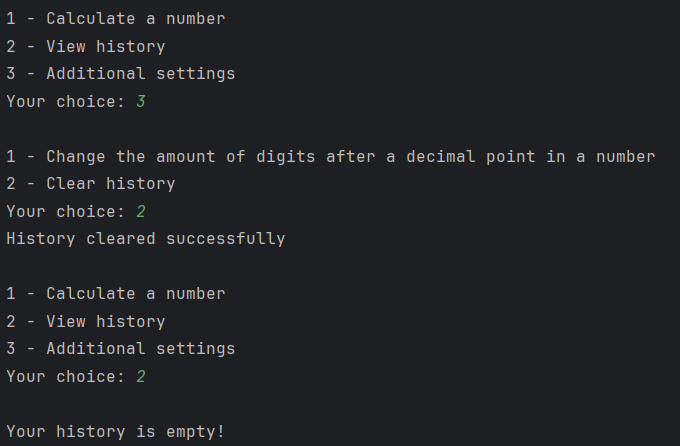
print("Invalid input")

На рис. 1-3 зображено результат виконання програми:



*Рис. 1. Приклад роботи калькулятора*

*  
Рис. 2. Приклад роботи функції історії обчислень*



*Рис. 3. Приклад роботи додаткових налаштувань*

*(у цьому прикладі функція очистки історії обчислень)*

Посилання на Github: [PaperGlit/Python\_Lab\_2](https://github.com/PaperGlit/Python_Lab_2)

**Висновок:**

Виконавши ці завдання, я перетворив консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. Цей проект допоміг мені вивчити концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.