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:

ст. гр. РІ-31

Ольга КЛІЩЕВСЬКА

Прийняв:

Сергій ЩЕРБАК

Львів-2024

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

**Хід роботи**

**Завдання:**

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

Реалізуйте метод \_\_init\_\_ у класі Calculator для ініціалізації необхідних атрибутів або змінних. Метод повинен приймати введення для двох чисел і 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. Реалізуйте обробку помилок у межах класу Calculator для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідні повідомлення про помилку. Покращте інтерфейс користувача у межах класу Calculator, надавши чіткі запити, повідомлення та форматування виводу для зручності читання.

from GlobalVariables import memory, history

class Calculator:

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

pass

def input\_numbers\_and\_operator(self):

try:

num1 = float(input("Enter the first number: ").replace(",", "."))

operator = input("Enter an operator (+, -, \*, /, %, sqrt, ^): ")

if operator != 'sqrt' and operator != '^':

num2 = float(input("Enter the second number: ").replace(",", "."))

else:

num2 = None

return num1, num2, operator

except ValueError:

print("Invalid input! Please enter numeric values.")

return None, None, None

def validate\_operator(self, operator):

valid\_operators = ['+', '-', '\*', '/', '%', 'sqrt', '^']

if operator in valid\_operators:

return True

else:

print("Error: Invalid operator.")

return False

def calculate(self, num1, num2, operator):

try:

if operator == '+':

return num1 + num2

elif operator == '-':

return num1 - num2

elif operator == '\*':

return num1 \* num2

elif operator == '/':

if num2 == 0:

return "Error: Division by zero is not allowed."

return num1 / num2

elif operator == '%':

if num2 == 0:

return "Error: Division by zero is not allowed."

return num1 % num2

elif operator == 'sqrt':

if num1 < 0:

return "Error: Negative number cannot have a square root."

return num1 \*\* 0.5

elif operator == '^':

return num1 \*\* 2

except Exception as e:

return f"Error: {e}"

def memory\_clear(self):

global memory

memory = 0

def memory\_recall(self):

global memory

return memory

def memory\_store(self, value):

global memory

memory = value

def memory\_add(self, value):

self.memory += value

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

global history

history.append(f"{expression} = {result}")

def view\_history(self):

global history

if not history:

print("No history available.")

else:

for entry in history:

print(entry)

def run(self):

while True:

print("\nMemory Operations: MC (Memory Clear), MR (Memory Recall), MS (Memory Store), M+ (Memory Add)")

print("Operator Options: +, -, \*, /, %, √, ^")

print("Type 'history' to view calculation history or 'exit' to quit.")

num1\_input = input("Enter the first number (or type 'history' or 'exit'): ").replace(",", ".")

if num1\_input.lower() == 'history':

self.view\_history()

continue

if num1\_input.lower() == 'exit':

break

try:

num1 = float(num1\_input)

except ValueError:

print("Invalid input! Please enter a valid number.")

continue

num2\_input = input("Enter the second number (or leave empty for square operation): ").replace(",", ".")

if num2\_input.lower() == 'history':

self.view\_history()

continue

if num2\_input.lower() == 'exit':

break

num2 = None

if num2\_input.strip() != "":

try:

num2 = float(num2\_input)

except ValueError:

print("Invalid input! Please enter a valid number.")

continue

operator = input("Enter an operator (+, -, \*, /, %, √, ^): ").strip()

if not self.validate\_operator(operator):

continue

result = self.calculate(num1, num2, operator)

expression = f"{num1} {operator} {num2 if num2 is not None else ''}"

self.add\_to\_history(expression, result)

print(f"Result: {result}")

memory\_command = input("Enter memory command (MC, MR, MS, M+) or press Enter to skip: ").strip().upper()

if memory\_command == 'MC':

self.memory\_clear()

print("Memory cleared.")

elif memory\_command == 'MR':

print("Memory recall:", self.memory\_recall())

elif memory\_command == 'MS':

self.memory\_store(result)

print("Memory stored:", memory)

elif memory\_command == 'M+':

self.memory\_add(result)

print("Memory updated:", memory)

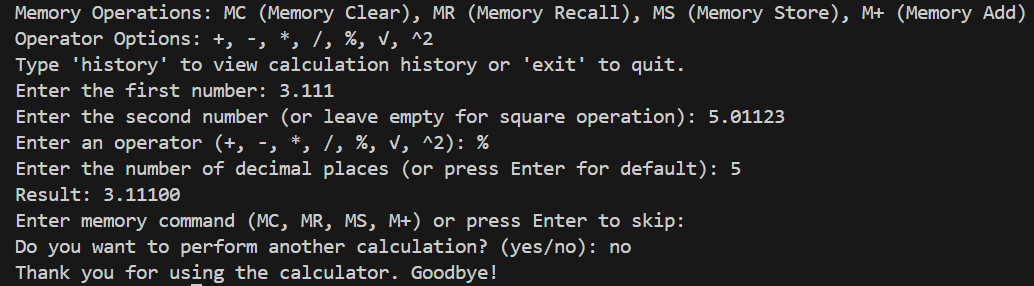
continue\_choice = input("Do you want to perform another calculation? (yes/no): ").strip().lower()

if continue\_choice != 'yes':

print("Thank you for using the calculator. Goodbye!")

break

На рис. 1 зображено результат роботи програми.



*Рис. 1. Результат виконання завдання 1*

Посилання на GitHub: https://github.com/olia28/SPL.git

**Висновок:** під час виконання лабораторної роботи, я перетворила 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.