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. ІТ-21сп

Олександр КОЗАК

Прийняв

доцент каф. ІСМ:

Сергій ЩЕРБАК

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Львів-2023

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

**Хід 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.

**Main.py**

from calculator import Calculator

class main():

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

calc = Calculator()

calc.runCalculator()

**calculator\_engine.py**

import math

class Calculator\_engine:

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

self.settings = {

"decimalPlaces": 2,

"useMemory": True

}

self.memory = None # Словник для збереження значень в пам'яті

self.history = [] # Список для збереження історії обчислень

# Метод для зберігання значення в пам'яті

def store\_in\_memory(self, value):

self.memory = value

# Метод для отримання значення з пам'яті

def recall\_from\_memory(self):

return self.memory

# Метод для додавання запису в історію обчислень

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

historyEntry = f'Вираз:{expression} Результат:{result}'

self.history.append(historyEntry)

# Метод для перегляду історії обчислень

def view\_history(self):

if not self.history:

print("Даних в істрії ще немає!")

for historyEntry in self.history:

print(historyEntry)

def change\_settings(self):

print("1. Зміна знаків після коми")

print("2. Налаштування функції пам'яті.")

choice = input("Введіть ваш вибір: ")

if choice == '1':

try:

places = int(input("Введіть читсло скільки знаків після коми повино бути (0-20): "))

if 0 <= places <= 20:

self.settings["decimalPlaces"] = places

print(f"Встановлено {places} знаків після коми!")

else:

print("Кількість знаків що ви ввели виходить за межі дозволеного.")

except ValueError:

print("Некоректний ввід.")

elif choice == '2':

self.settings["useMemory"] = not self.settings["useMemory"]

status = "enabled" if self.settings["useMemory"] else "disabled"

print(f"Статус функції пам'яті {status}.")

def calculateValue (self,firstNumber, secondNumber,operator):

try:

if operator == '-':

return firstNumber - secondNumber

if operator == '+':

return firstNumber + secondNumber

if operator == '\*':

return firstNumber \* secondNumber

if operator == '%':

return firstNumber % secondNumber

if operator == '^':

return firstNumber \*\* secondNumber

if operator == '√':

return math.sqrt(firstNumber)

if operator == '/':

if secondNumber == 0: # Перевірка чи значення не нуль

print("Помилка: Ділення на нуль!")

return None

return firstNumber / secondNumber

except Exception as e:

print(f"Виникла помилка: {e}")

return None

def getParamValue(self, inputMessage):

while True:

try:

param = float(input(inputMessage))

return param

except ValueError:

print("Невіртний формат числа, Будь ласк, ввдедіть число знову")

def getOperator(self, inputOperatorMasseg):

while True:

operator = input(inputOperatorMasseg)

if operator in ('+', '-', '\*', '/', '%', '^', '√'):

return operator

else:

print("Неправильний оператор. Спробуйте ще раз.")

def userMenu(self):

print("\nМеню калькулятора:")

print("1. Перейти до обчислення виразу.")

print("2. Змінити налаштування.")

print("3. Переглянути історію.")

print("4. Вихід.")

**calculator.py**

from calculator\_engine import Calculator\_engine

class Calculator(Calculator\_engine):

def runCalculator(self):

while True:

self.userMenu()

main\_menu\_choice = input("Enter your choice: ")

if main\_menu\_choice == "1":

if self.memory is not None:

print(f"Number in memory {self.memory}")

get\_param\_from\_memory = input("Take the number from memory? Y/N \n").strip().lower()

if get\_param\_from\_memory == 'y':

first\_param = self.memory

else:

first\_param = self.getParamValue("Enter first param:\n")

else:

first\_param = self.getParamValue("Enter first param:\n")

operator = self.getOperator("Enter the operator from the ones presented (+, -, \*, /, %, ^, √): ")

if operator == '√':

result\_of\_calculation = round(self.calculateValue(first\_param, None, operator),self.settings["decimalPlaces"])

expression = f" {operator}{first\_param}"

print("Result of expression:\n",operator, first\_param, "=",result\_of\_calculation)

else:

second\_param = self.getParamValue("Enter second param:\n")

result\_of\_calculation = round(self.calculateValue(first\_param, second\_param, operator),self.settings["decimalPlaces"])

expression = f"{first\_param} {operator} {second\_param}"

print("Result of expression:\n", first\_param, operator, second\_param, "=", result\_of\_calculation)

# Adding the result to the calculation history

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

if self.settings["useMemory"]:

self.store\_in\_memory(result\_of\_calculation)

elif main\_menu\_choice == '2':

self.change\_settings()

elif main\_menu\_choice == '3':

self.view\_history()

elif main\_menu\_choice == '4':

break

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

*Рис. 1 Результат виконання програми*

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