Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра ІСМ

Звіт
про виконання лабораторної роботи № 2
«Основи побудови об’єктно-орієнтованих додатків на Python»
з дисципліни
«Спеціалізовані мови програмування»

Виконав:
Студент групи ІТ-32,
Вольвенко І. Р.

Прийняв:
Щербак С.С

Львів 2023

Мета роботи:

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

Завдання:

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:

runner.py:

```
from calculator import Calculator
from console_output import ConsoleOutput

if __name__ == "__main__":
    decimal = int(ConsoleOutput.get_user_input("Введіть скільки десяткових розрядів після коми показувати: "))
    max_history_size = int(ConsoleOutput.get_user_input("Введіть скільки обчислень зберігати у історії: "))

    calculator = Calculator(decimal, max_history_size)

    while True:
```

```
        try:
            ConsoleOutput.show_message("\nМеню:")
            ConsoleOutput.show_message("R - Відновлення результату")
            ConsoleOutput.show_message("H - Історія обчислень")
            ConsoleOutput.show_message("K - Використати калькулятор")

            choice = ConsoleOutput.get_user_input("Ваш вибір: ").lower()

            if choice == 'k':
                num1 = float(ConsoleOutput.get_user_input("Введіть перше число: "))
                operator = ConsoleOutput.get_user_input("Введіть оператор (+, -, *, /, ^, √, %): ")

                if operator not in ['+', '-', '*', '/', '^', '√', '%']:
                    raise ValueError("Недійсний оператор. Введіть один із +, -, *, /, ^, √, %")

                if operator != '√':
                    num2 = float(ConsoleOutput.get_user_input("Введіть друге число: "))
                    result = calculator.calculate(num1, operator, num2)
                else:
                    result = calculator.calculate(num1, operator, 0)

                ConsoleOutput.show_message(f"Результат: {result}")

                save_quest = ConsoleOutput.get_user_input("Хочете зберегти результат?(y/n): ").lower()
                if save_quest == 'y':
                    calculator.save_result(num1, operator, num2, result)

            elif choice == 'h':
                ConsoleOutput.show_message("\nІсторія обчислень:")
                for item in calculator.display_history():
                    ConsoleOutput.show_message(item)

            elif choice == 'r':
                ConsoleOutput.show_message("\nЗбережені обчислення:")
```

```
            for item in calculator.display_memory():
                ConsoleOutput.show_message(item)

        except ValueError as e:
            ConsoleOutput.show_message(f"Помилка: {e}")
        except ZeroDivisionError as e:
            ConsoleOutput.show_message(f"Помилка: {e}")

        another_calculation = ConsoleOutput.get_user_input("Бажаєте виконати ще одне обчислення? (y/n): ")
        if another_calculation.lower() != 'y':
            break
```

calculator.py:

```
class Calculator:
    def __init__(self, decimal=2, max_history_size=10):
        self.memory = []
        self.history = []
        self.decimal = decimal
        self.max_history_size = max_history_size
        self.result = 0
        self.num1 = 0
        self.operator = ''
        self.num2 = 0

    def calculate(self, num1, operator, num2):
        result = 0

        if operator == '+':
            result = num1 + num2
        elif operator == '-':
            result = num1 - num2
        elif operator == '*':
            result = num1 * num2
```

```
        elif operator == '/':
            if num2 == 0:
                raise ZeroDivisionError("Ділення на нуль недопустимо.")
            result = num1 / num2
        elif operator == '^':
            result = num1 ** num2
        elif operator == '√':
            result = num1 ** 0.5
        elif operator == '%':
            result = num1 % num2

        result = round(result, self.decimal)
        self.update_history(num1, operator, num2, result);
        return result

    def save_result(self, num1, operator, num2, result):
        self.memory.append(f"{num1} {operator} {num2} = {result}")

    def display_memory(self):
        return self.memory

    def display_history(self):
        return self.history

    def update_history(self, num1, operator, num2, result):
        if len(self.history) >= self.max_history_size:
            del self.history[0]

        self.history.append(f"{num1} {operator} {num2} = {result}")
```

console_output.py:

```
class ConsoleOutput:
    @staticmethod
    def show_message(message):
        print(message)
```

```
    @staticmethod
    def get_user_input(prompt):
        return input(prompt)
```

Посилання на GitHub репозиторій: https://github.com/Deadmarvald/smp

Висновки: 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.