МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ



ЗВІТ

Про виконання лабораторної роботи №2

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

З дисципліни

«Спеціалізовані мови програмування»

Студентки групи РІ – 31

Нащецької Яни Вікторівни

Прийняв викладач

Щербак С.С.

Лабораторна робота № 2. Основи побудови об’єктно-орієнтованих додатків на Python.

**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.

**Основний код програми:**

"""

Модуль `calculator.py` реалізує клас `Calculator`, який забезпечує базові функції калькулятора,

такі як арифметичні операції, управління пам'яттю та ведення історії.

"""

import shared.functions.manage\_app\_settings as manage\_app\_settings

from shared.constants.global\_variables import Button, operations

from shared.logger import Logger

class Calculator:

    """

    Клас `Calculator` забезпечує функціональність для виконання математичних операцій,

    управління пам'яттю та ведення історії обчислень.

    Атрибути:

        current\_value (float): Поточне значення калькулятора.

        memory (float): Значення, збережене в пам'яті калькулятора.

        history (list): Список, який містить історію виконаних операцій.

        decimal\_places (int): Кількість знаків після коми для відображення результату.

    """

    def \_\_init\_\_(self, current\_value=0, memory=0):

        """

        Ініціалізує екземпляр класу Calculator.

        Args:

            current\_value (float): Початкове значення калькулятора (за замовчуванням 0).

            memory (float): Початкове значення пам'яті калькулятора (за замовчуванням 0).

        """

        self.current\_value = current\_value

        self.memory = memory

        self.history = []

        self.decimal\_places = manage\_app\_settings.decimal\_places

    def validate\_number(self, value):

        """

        Перевіряє, чи є введене значення числом.

        Args:

            value: Значення для перевірки.

        Returns:

            float: Перетворене числове значення.

        Raises:

            ValueError: Якщо введене значення не є числом.

        """

        try:

            return float(value)

        except ValueError:

            raise ValueError(f"Помилка. Введене значення не є числом")

    def validate\_operator(self, operator):

        """

        Перевіряє, чи є введений оператор коректним.

        Args:

            operator (str): Оператор для перевірки.

        Returns:

            str: Перевірений оператор.

        Raises:

            ValueError: Якщо оператор некоректний.

        """

        if not operator in {button.value for button in Button}:

            raise ValueError(f"Введено некоректний оператор.")

        return operator

    def calc(self, number\_1, operator, number\_2=None):

        """

        Виконує математичну операцію на основі введених чисел та оператора.

        Args:

            number\_1 (float): Перше число.

            operator (str): Оператор для виконання операції.

            number\_2 (float, optional): Друге число (для унарних операторів не потрібне).

        Returns:

            float: Результат операції.

        Raises:

            ValueError: Якщо операція не може бути виконана.

        """

        try:

            if operator == Button.SQRT.value:

                Logger.log(f"{number\_1} {operator} {number\_2}")

                return operations[operator](number\_1)

            else:

                return operations[operator](number\_1, number\_2)

        except Exception as e:

            raise ValueError(f"Помилка виконання операції: {e}")

    def add\_to\_history(self, param\_1, operator, param\_2, result):

        """

        Додає виконану операцію до історії.

        Args:

            param\_1 (float): Перше число.

            operator (str): Використаний оператор.

            param\_2 (float): Друге число.

            result (float): Результат виконаної операції.

        """

        if operator == Button.SQRT:

            self.history.append(f"{operator} {param\_1} = {result}")

        self.history.append(f"{param\_1} {operator} {param\_2} = {result}")

    def add\_to\_memory(self, param):

        """

        Додає число до пам'яті калькулятора.

        Args:

            param (float): Число, яке потрібно додати до пам'яті.

        Raises:

            ValueError: Якщо введене значення не є числом.

        """

        number = self.validate\_number(param)

        print(number + self.memory)

"""

Модуль для запису в консоль виведення результатів калькулятора.

Визначає функції для відображення інформації користувачеві.

"""

from shared.functions.ui\_functions.handle\_input import write\_to\_output

class ConsoleWriter:

    """

    Клас для запису інформації в консоль.

    Містить методи для виведення результатів обчислень користувачу.

    """

    def output\_value(self, value, message="Ваше значення"):

        """

        Виводить результат в консоль.

        Параметри:

            output (str): Результат обчислення для виведення.

        """

        write\_to\_output(f"{message}: {value}")

import shared.functions.manage\_app\_settings as manage\_app\_settings

from shared.classes.calculator.сonsole\_reader import ConsoleReader

from shared.classes.calculator.calculator import Calculator

from shared.classes.calculator.console\_writer import ConsoleWriter

from shared.constants.global\_variables import main\_menu, memory\_menu, settings\_menu

from shared.functions.ui\_functions.menu\_functions import print\_menu

class CalculatorConsole:

    def \_\_init\_\_(self):

        self.consoleWriter = ConsoleWriter()

        self.consoleReader = ConsoleReader()

        self.calculator = Calculator()

    def run\_console(self):

        while True:

            print\_menu(manage\_app\_settings.console\_color, main\_menu)

            answer = self.consoleReader.input\_data("Оберіть варіант меню")

            match answer:

                case 1:

                    self.handle\_calc()

                case 2:

                    self.consoleWriter.output\_value(

                        self.calculator.current\_value, "Відповідь"

                    )

                case 3:

                    self.consoleWriter.output\_value(self.calculator.history, "Історія")

                case 4:

                    self.handle\_settings\_menu()

                case 5:

                    self.handle\_memory\_menu()

                case 6:

                    break

                case \_:

                    self.consoleWriter.output\_value(

                        answer, "Такого варіанта в меню немає"

                    )

            continue\_answer = self.consoleReader.read\_value("Продовжити? (так/ні)")

            if continue\_answer == "ні":

                break

    def handle\_calc(self):

        param\_1 = self.consoleReader.input\_data("Введіть перше число")

        operator = self.consoleReader.input\_data("Введіть оператор", "operator")

        param\_2 = self.consoleReader.input\_data("Введть друге число")

        result = self.calculator.calc(param\_1, operator, param\_2)

        result = round(result, int(manage\_app\_settings.decimal\_places))

        self.calculator.current\_value = result

        self.calculator.add\_to\_history(param\_1, operator, param\_2, result)

    def handle\_settings\_menu(self):

        while True:

            print\_menu(manage\_app\_settings.console\_color, settings\_menu)

            option = self.consoleReader.input\_data("Оберіть варіант меню")

            match option:

                case 1:

                    new\_decimal = self.consoleReader.input\_data(

                        "Введіть нову кількість знаків після коми"

                    )

                    manage\_app\_settings.set\_decimal(new\_decimal)

                    self.consoleWriter.output\_value(

                        manage\_app\_settings.decimal\_places,

                        "Кількість знаків після коми змінено на",

                    )

                case 2:

                    self.consoleWriter.output\_value(

                        manage\_app\_settings.console\_color, "Поточний колір консолі"

                    )

                    new\_color = self.consoleReader.input\_data(

                        """Введіть новий колір.

                            Доступні кольори:

                            black, red, green, yellow, blue, magenta, cyan, white

                        """,

                        "color",

                    )

                    manage\_app\_settings.set\_console\_color(new\_color)

                    self.consoleWriter.output\_value(

                        manage\_app\_settings.console\_color, "Колір консолі змінено на"

                    )

                case 3:

                    break

                case \_:

                    self.consoleWriter.output\_value(

                        option, "Такого ваіанта в меню немає"

                    )

    def handle\_memory\_menu(self):

        while True:

            print\_menu(manage\_app\_settings.console\_color, memory\_menu)

            option = self.consoleReader.input\_data("Оберіть варіант меню")

            match option:

                case 1:

                    self.calculator.memory = 0

                case 2:

                    self.consoleWriter.output\_value(

                        self.calculator.memory, "Поточне значення"

                    )

                case 3:

                    self.calculator.memory = self.consoleReader.input\_data(

                        "Введіть нове значення пам'яті"

                    )

                case 4:

                    new\_value = self.consoleReader.input\_data(

                        "Додати в пам'ять таке значення"

                    )

                    self.calculator.memory += new\_value

                case 5:

                    break

                case \_:

                    self.consoleWriter.output\_value(

                        option, "Такого ваіанта в меню немає"

                    )

from shared.functions.ui\_functions.handle\_input import read\_from\_input

from shared.functions.validation\_functions import (

    check\_color,

    check\_number,

    check\_operator,

)

class ConsoleReader:

    def read\_value(self, message=""):

        return read\_from\_input(f"{message}: ")

    def check\_value(self, value, value\_type="number"):

        match value\_type:

            case "number":

                return check\_number(value)

            case "operator":

                return check\_operator(value)

            case "color":

                return check\_color(value)

            case \_:

                raise ValueError(f"Невідомий тип: '{value\_type}'.")

    def input\_data(self, message="Введіть значення", type="number"):

        param = self.read\_value(message)

        param = self.check\_value(param, type)

        return param

**Висновок**: за допомогою цієї 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.