МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра інформаційних систем та мереж

ЗВІТ

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

Виконала:
ст. гр. ІТ-32,
Троцько О. М.
Прийняв:
Щербак С. С.

ЛЬВІВ – 2023

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.

Код програми:

```
class Calculator:

    def __init__(self):
        self.history = []
        self.decimal_places = 2


    def check_operator(self, operator):
        if operator in ['+', '-', '*', '/', '**', '//', '%']:
            return True
        else:
            return False

    def get_parameters_and_operator(self):
```

```
        parameter1 = float(input("Enter the first number: "))
        parameter2 = float(input("Enter the second number: "))
        operator = input("Enter the operator ['+', '-', '*', '/', '**', '//', '%']:
")

        if not self.check_operator(operator):
            raise ValueError("The operator is invalid!")

        return parameter1, parameter2, operator

    def count_sum(self, parameter1, parameter2):
        return parameter1 + parameter2

    def count_difference(self, parameter1, parameter2):
        return parameter1 - parameter2

    def count_product(self, parameter1, parameter2):
        return parameter1 * parameter2

    def count_quotient(self, parameter1, parameter2):
        if parameter2 == 0:
            raise ZeroDivisionError("Division by zero is not allowed.")
        return parameter1 / parameter2

    def count_power(self, parameter1, parameter2):
        return parameter1 ** parameter2

    def count_square_root(self, parameter1, parameter2):
        if parameter1 < 0 or parameter2 < 0:
            raise ValueError("Square root of negative number is not allowed.")
        return parameter1 ** 0.5, parameter2 ** 0.5

    def count_remainder(self, parameter1, parameter2):
        if parameter2 == 0:
            raise ZeroDivisionError("Division by zero is not allowed.")
        return parameter1 % parameter2

    def make_calculations(self, parameter1, parameter2, operator):
        match operator:
            case "+":
```

```
            return self.count_sum(parameter1, parameter2)
        case "-":
            return self.count_difference(parameter1, parameter2)
        case "*":
            return self.count_product(parameter1, parameter2)
        case "/":
            return self.count_quotient(parameter1, parameter2)
        case "**":
            return self.count_power(parameter1, parameter2)
        case "//":
            return self.count_square_root(parameter1, parameter2)
        case "%":
            return self.count_remainder(parameter1, parameter2)


def set_decimal_places(self, places):
    self.decimal_places = places

def add_to_history(self, result):
    self.history.append(result)
    return self.history

def view_history(self):
    for i, index in self.history:
        print(index, "-", i)

def clear_history(self):
    self.history = []

def menu(self):
    while True:
        print("Menu:")
        print("1 - Settings")
        print("2 - Make Calculations")
        print("3 - View History")
        print("0 - Exit")

        choice = str(input("Enter your choice: "))
        match choice:
            case "1":
```

```
                    print("Settings:")
                    print("1 - Change Decimal Places (default 2)")
                    print("2 - Clear History")
                    sub_choice = input("Enter your setting choice: ")

                    if sub_choice == "1":
                            self.set_decimal_places(int(input("Enter the number of
decimal places: ")))
                    elif sub_choice == "2":
                        self.clear_history()
                        print("History was cleaned!")
                case "2":
                                            parameter1, parameter2, operator =
self.get_parameters_and_operator()
                           calculation_result = self.make_calculations(parameter1,
parameter2, operator)
                    if operator == "//":
                         formatted_calculation_result1 = f"{calculation_result[0]:.
{self.decimal_places}f}"
                         formatted_calculation_result2 = f"{calculation_result[1]:.
{self.decimal_places}f}"
                                        result = f"Square root of {parameter1} =
{formatted_calculation_result1}, Square root of {parameter2} =
{formatted_calculation_result2}"
                    else:
                            formatted_calculation_result = f"{calculation_result:.
{self.decimal_places}f}"
                               result = f"{parameter1} {operator} {parameter2} =
{formatted_calculation_result}"

                    print(result)

                     sub_choice = str(input("Do you want to save the result? [Y/n]
"))
                     if sub_choice == "y" or sub_choice == "Y" or sub_choice == "\
n":
                        self.add_to_history(result)
                    else:
                        print("The result wasn't saved")
```

```
                case "3":
                    if self.history:
                        self.view_history()
                    else:
                        print("No calculation history yet.")
                case "0":
                    break
                case _:
                    print("Invalid option! Try again!")

from Calculator import Calculator

def __main__():
    calculator = Calculator()
    calculator.menu()


if __name__ == "__main__":
    __main__()
```

GitHub Repository: https://github.com/trolchiha/SPL-labs.git

Висновок: під час виконання лабораторної роботи навчилась перетворювати консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python.