Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Лабораторная работа №4

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-32Б
Поддубный М.Н.

Проверил:
Канев А.И.

2021 г.

Описание задания

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать следующий каталог. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. Вместо реализации паттерна Вы можете написать тесты для своей программы решения биквадратного уравнения. В этом случае, возможно, Вам потребуется доработать программу решения биквадратного уравнения, чтобы она была пригодна для модульного тестирования.
3. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.
    - Создание Mock-объектов.

Текст программы

main.py

```
import sys
import math


def get_coef(index, prompt):
    '''
 Читаем коэффициент из командной строки или вводим с клавиатуры
 Args:
 index (int): Номер параметра в командной строке
 prompt (str): Приглашение для ввода коэффицента
 Returns:
 float: Коэффициент квадратного уравнения
    '''
    ass = True
    while ass:
        try:
            # Пробуем прочитать коэффициент из командной строки
            print(prompt)
            coef_str = input()
            ass1 = float(coef_str)
            ass = False
        except:
            # Вводим с клавиатуры
            ass = True
            print('Повторите ввод коэффииента')
    # Переводим строку в действительное число
    coef = float(coef_str)
    return coef


def get_roots(a, b, c):
    '''
 Вычисление корней квадратного уравнения
 Args:
 a (float): коэффициент A
```

```
 b (float): коэффициент B
 c (float): коэффициент C
 Returns:
 list[float]: Список корней
    '''
    result = []
    if a == 0 and b == 0 and c == 0:
        return result
    if a == 0 and b == 0:
        return result
    if a == 0 and b != 0:
        if c <= 0:
            D1 = - 4 * b * c
            sqD1 = math.sqrt(D1)
            root1 = sqD1 / (2 * b)
            if root1 == 0:
                result.append(abs(root1))
            else:
                result.append(root1)
            if root1 != -root1:
                result.append(-root1)
        else:
            return result
    if a != 0:
        D = b * b - 4 * a * c
        if D == 0.0:
            root = -b / (2.0 * a)
            if root >= 0:
                sqrt = math.sqrt(root)
                result.append(sqrt)
                if sqrt != 0:
                    result.append(-sqrt)
        elif D > 0.0:
            sqD = math.sqrt(D)
            root3 = (-b + sqD) / (2.0 * a)
            root4 = (-b - sqD) / (2.0 * a)
            root5 = (-b + sqD) / (2.0 * a)
            root6 = (-b - sqD) / (2.0 * a)
            if root3 >= 0:
                sqrt3 = math.sqrt(root3)
                result.append(sqrt3)
                sqrt5 = -math.sqrt(root5)
                if sqrt3 != sqrt5:
                    result.append(sqrt5)
            if root4 >= 0:
                sqrt4 = math.sqrt(root4)
                result.append(sqrt4)
                sqrt6 = -math.sqrt(root6)
                if sqrt4 != sqrt6:
                    result.append(sqrt6)
    return result


def main():
    '''
    Основная функция
    '''
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')
    # Вычисление корней
    roots = get_roots(a, b, c)
    # Вывод корней
    len_roots = len(roots)
```

```
    if a == b == c == 0:
        print('Бесконеное число корней')
    elif len_roots == 0:
        print('Нет корней')
    elif len_roots == 1:
        print('Один корень: {}'.format(roots[0]))
    elif len_roots == 2:
        print('Два корня: {} и {}'.format(roots[0], roots[1]))
    elif len_roots == 3:
        print('Три корня: {} и {} и {}'.format(roots[0], roots[1], roots[2]))
    elif len_roots == 4:
        print('Четыре корня: {} и {} и {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))


# Если сценарий запущен из командной строки
if __name__ == "__main__":
    main()

# Пример запуска
# qr.py 1 0 -4
```

testing.py

```
import unittest
import sys, os
import math
sys.path.append(os.getcwd())
from main import *


class Testget_roots(unittest.TestCase):
    def test_get_roots(self):
        self.assertEqual(get_roots(0, 0, 0), [])
        self.assertEqual(get_roots(5, 6, 7), [])
        self.assertEqual(get_roots(0, 4, 0), [0])
        self.assertEqual(get_roots(1, -18, 81), [3, -3])
        self.assertEqual(get_roots(1, -4, 0), [2, -2, 0])
        self.assertEqual(get_roots(1, -5, 6), [math.sqrt(3), -math.sqrt(3),
math.sqrt(2), -math.sqrt(2)])


if __name__ == '__main__':
    unittest.main()
```

mock.py

```
import unittest
from main import get_roots
from unittest.mock import patch, Mock


class Testget_roots(unittest.TestCase):
    @patch('main.get_roots', return_value = [])
    def test_get_roots(self, a):
        self.assertEqual(set(get_roots(5, 6, 7)), set([]))

    @patch('main.get_roots', return_value=[0])
    def test_get_roots(self, a):
        self.assertEqual(set(get_roots(0, 4, 0)), set([0]))

    @patch('main.get_roots', return_value=[3, -3])
    def test_get_roots(self, a):
        self.assertEqual(set(get_roots(1, -18, 81)), set([3, -3]))
```

BDD.py

```
from behave import *
from main import *


@given('I have 3 koef')
def coef(step, n1, n2, n3):
    step.context.n1 = n1
    step.context.n2 = n2
    step.context.n3 = n3


@when('I count the roots')
def roots(step):
    step.context.result = get_roots(step.context.n1, step.context.n2,
step.context.n3)


@then('checking the result')
def c(step, result):
    assert step.context.result == result
```

test.feature.py

```
Feature: BDD Test

  Scenario: Test
    Given I have 3 koef
     When I count the roots
     Then checking the result
```

Экранные формы с результатами выполнения программы

```
Ran 1 test in 0.002s

OK
```

```
Ran 1 test in 0.002s

OK
```

```
Ran 1 test in 0.002s

OK
```