Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчет по лабораторной работе №4

Выполнил:
студент группы ИУ5-31Б
Рысьева Елизавета
Антоновна

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Лабораторная работа №4
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 getcoeffromkeyboard(prompt):
    while True:
        try:
            print(prompt)
            coef = float(input())
        except ValueError:
            print('Коэффициент введен неверно, попробуйте еще раз')
        else:
            break
    return coef


def getcoef(index, prompt):
    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
        try:
            coef = float(coef_str)
        except ValueError:
            print('Коэффициент введен неверно, попробуйте еще раз')
            coef = getcoeffromkeyboard(prompt)
    except:
        # Вводим с клавиатуры
        coef = getcoeffromkeyboard(prompt)
    return coef
```

```
def getrootsallcoef(a, b, c):
    result = []
    D = b * b - 4 * a * c
    if D == 0.0:
        root = -b / (2.0 * a)
        if root > 0:
            result.append(math.sqrt(root))
            result.append(math.sqrt(-root))
        if root == 0:
            result.append(0.0)
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)
        if root1 > 0:
            result.append(math.sqrt(root1))
            result.append(-math.sqrt(root1))
        if root1 == 0:
            result.append(0.0)
        if root2 > 0:
            result.append(math.sqrt(root2))
            result.append(-math.sqrt(root2))
        if root2 == 0:
            result.append(0.0)
    return result


def getroots(a, b, c):
    if a == 0:
        if b == 0:
            if c == 0:
                return ['inf']
            else:
                return []
        else:
            if c == 0:
                return [0]
            else:
                return [-c / b]
    else:
        return getrootsallcoef(a, b, c)


def main():
    a = getcoef(1, 'Введите коэффициент A:')
    b = getcoef(2, 'Введите коэффициент B:')
    c = getcoef(3, 'Введите коэффициент C:')
    # Вычисление корней
    roots = getroots(a, b, c)
    # Вывод корней
    len_roots = len(roots)
    if len_roots == 0:
        print('Нет корней')
    elif len_roots == 1:
        if roots[0] == 'inf':
            print('Бесконечное множество корней')
        else:
            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 {}'.format(roots[0], roots[1], roots[2]))
    elif len_roots == 4:
```

```
        print('Четыря корня: {}, {}, {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))
```

teststdd.py

```
import main
import unittest
from unittest import mock


class Tests(unittest.TestCase):

    def test_chetire_kornya(self):
        roots = main.getroots(4, -5, 1)
        self.assertEqual([1, -1, 0.5, -0.5], roots)

    def test_tri_kornya(self):
        roots = main.getroots(-1, 4, 0)
        self.assertEqual([0, 2, -2], roots)

    def test_dva_kornya(self):
        roots = main.getroots(-2, 0, 10)
        self.assertAlmostEqual(1.495, roots[0], 3)
        self.assertAlmostEqual(-1.495, roots[1], 3)

    def test_nol_korney(self):
        roots = main.getroots(1, 2, 3)
        self.assertEqual([], roots)

    @mock.patch('main.getroots', return_value=[322])
    def test_mock(self, get_roots):
        self.assertEqual(main.getroots(1, 2, 3), [322])
```

bdd.feature

```
Feature: chetire kornya
  Scenario: korni x^4 2x^2 -1
    Given I have 1*x^4 + -17*x^2 + 16 = 0
    When I solve this equation
    Then I expect to get four korney: 4.0, -4.0, 1.0, -1.0
```

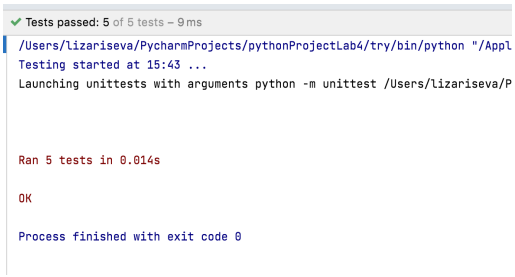
steps.py

```
from main import *
from behave import given, when, then


@given(u'I have {a}*x^4 + {b}*x^2 + {c} = 0')
def step_impl(context, a: float, b: float, c: float):
    context.a = float(a)
    context.b = float(b)
    context.c = float(c)


@when(u'I solve this equation')
def step_impl(context):
    context.roots = getroots(context.a, context.b, context.c)


@then(u'I expect to get four korney: {x1}, {x2}, {x3}, {x4}')
```

```python
def step_impl(context, x1: float, x2: float, x3: float, x4: float):
    result = [float(x1), float(x2), float(x3), float(x4)]
    assert context.roots == result
```

Примеры выполнения программы

tests.py



behave