Московский государственный технический университет им. Н.Э. Баумана

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

Отчет по лабораторной работе №4
«Шаблоны проектирования и модульное тестирование Python»

Выполнил: Ким Алексей Максимович ИУ5-32Б
Дата: 20.12.2021

Москва, 2021 г.

Постановка задачи:

Цель лабораторной работы: изучение реализации шаблонов проектирования и возможностей модульного тестирования в языке Python.

Требования к отчету:
Отчет по лабораторной работе должен содержать:
1. титульный лист;
2. описание задания;
3. текст программы;
4. экранные формы с примерами выполнения программы.

Задание:
- 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.
- 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.
- В модульных тестах необходимо применить следующие технологии:

TDD - фреймворк.
BDD - фреймворк.
Создание Mock-объектов.

Текст программ:

Main.py

```
import math


def vvod_koef():
    print('Введите коэффициенты A B C: ')
    try:
        a, b, c = map(int, input().split())
        return a, b, c
    except ValueError:
        print('Не корректно введены коэффициенты')
        a, b, c = vvod_koef()
        return a, b, c


def disc(a, b, c):
    d1 = b * b - 4 * a * c
    if d1 < 0:
        noans()
        return False
    else:
        return math.sqrt(d1)


def aravnonull(b, c):
    if c * b < 0:
        x1 = math.sqrt(-c / b)
        print(f'Корни равны -{x1} {x1}', end=' ')
        return True
    else:
        return False


def korni(a, b, d):
    x1, x2 = (-b + d) / (2 * a), (-b - d) / (2 * a)
    return x1, x2


def noans():
    print("Нет решений")
def get_roots(a, b, c):
    result = []
```

```
    if a != 0 and c != 0:
        d = disc(a, b, c)
        if d:
            x1, x2 = korni(a, b, d)
            if x1 < 0 and x2 > 0:
                x2 = math.sqrt(x2)
                # print(f"Корнями уравнения является -{x2} {x2
                result.append(-x2)
                result.append(x2)
            elif x2 < 0 and x1 > 0:
                x1 = math.sqrt(x1)
                result.append(-x1)
                result.append(x1)
                # print(f"Корнями уравнения является -{x1} {x1
            elif x1 > 0 and x2 > 0:
                x1, x2 = map(math.sqrt, [x1, x2])
                result.append(x1)
                result.append(-x1)
                result.append(x2)
                result.append(-x2)
                # print(f'Корнями уравнения являются -{x1} {x1
            else:
                noans()
        elif d == 0:
            if b > 0:
                noans()
            else:
                x1 = math.sqrt(-b / 2 * a)
                # print(f'Корниями уравнения являются -{x1} {x
                result.append(-x1)
                result.append(x1)
    # elif a == 0 and b == 0 and c == 0:
    #     print("Решением являются все действительные числа")
    elif a == 0 and c == 0:
        print('Решением является 0')
        result.append(0)
    elif c == 0:
        if not aravnonull(a, b):
            pass
        #noans()
        else:
```

```
            result.append(0)
    # else:
    #     if not aravnonull(b, c):
    #         noans()
    return result
if __name__ == "__main__":
    a, b, c = vvod_koef()
    print(get_roots(a, b, c))
```

Test_main.py

```
import unittest
from unittest.mock import patch, Mock
from main import disc
import math

class TestRoots(unittest.TestCase):

    def test_roots_is_equal(self):
        self.assertEqual(disc(1, -5, 6), 1.0)
        self.assertEqual(disc(1, -4, 4), 0.0)
        self.assertEqual(disc(-4, 16, 0), 16.0)
        self.assertEqual(disc(1, 0, -16), 8.0)

    def test_string_root(self):
        self.assertRaises(TypeError, disc, 'abobas')
        self.assertRaises(TypeError, disc, False)
        self.assertRaises(TypeError, disc, [3, 1])

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

Test_bdd.feature

```
Feature: Test Biquadratic equation Functionality

Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "-5", and "6"
    Then I get result "1.7320508075688772, -1.7320508075688772, 1.4142135623730951, -1.4142135623730951"

Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "-4", and "4"
    Then I get result "-1.4142135623730951, 1.4142135623730951"

Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "0", and "-16"
    Then I get result "-2.0, 2.0"
```

Bdd_test.py

```python
from behave import given, when, then
from main import get_roots


@given(u"Biquadratic equation app is run")
def step_impl(context):
    print(u'Step: Given Biquadratic equation app is run')

@when(u'I have the odds "{a}", "{b}", and "{c}"')
def step_impl(context, a, b, c):
    print(f'Step: I have the odds "{a}", "{b}", and "{c}"')
    b = str(get_roots(int(a), int(b), int(c))).rpartition(']')[0]
    c = b.partition('[')[2]
    context.result = c
    print(f'Stored result "{context.result}" in context')

@then(u'I get result "{out}"')
def step_impl(context, out):
    if (context.result == str(out)):
        print(f'Step: Then I get right result "{context.result}", "{out}"')
        pass
    else:
        raise Exception("Invalid root is returned.")
```

Вывод программ:

```
0 features passed, 1 failed, 0 skipped
1 scenario passed, 2 failed, 0 skipped
7 steps passed, 2 failed, 0 skipped, 0 undefined
Took 0m0.002s
(venv) alex@MacBook-Pro-Alex features % behave
Feature: Test Biquadratic equation Functionality # test_bdd.feature:1

  Scenario: Test my biquadratic equation
    Given Biquadratic equation app is run
    When I have the odds "1", "-5", and "6"
    Then I get result "1.7320508075688772, -1.7320508075688772, 1.4142135623730951, -1.4142135623730951"

  Scenario: Test my biquadratic equation                      # test_bdd.feature:8
    Given Biquadratic equation app is run                     # ../steps/bdd_test.py:6 0.000s
    When I have the odds "1", "-4", and "4"                   # ../steps/bdd_test.py:10 0.000s
    Then I get result "-1.4142135623730951, 1.4142135623730951" # ../steps/bdd_test.py:18 0.000s

  Scenario: Test my biquadratic equation     # test_bdd.feature:13
    Given Biquadratic equation app is run    # ../steps/bdd_test.py:6 0.000s
    When I have the odds "1", "0", and "-16" # ../steps/bdd_test.py:10 0.000s
    Then I get result "-2.0, 2.0"            # ../steps/bdd_test.py:18 0.000s

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
```