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 ИУ5-34Б
Барабанщиков Лев

Подпись и дата: 23.12.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г.

Постановка задачи

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.

Текст программы (main.py)

```
from sys import argv
from math import sqrt


def get_coef(index, msg):
    try:
        coef = float(argv[index])
    except:
        switch = True
        while switch:
            try:
                coef = float(input(msg))
                switch = False
            except:
                switch = True
    return coef


def discriminant(a, b, c):
    return b**2 - 4*a*c


def solve_square(a, b, c):
    roots = set()
    d = discriminant(a, b, c)
    if d >= 0:
        roots.add((-b-sqrt(d))/(2*a))
        roots.add((-b+sqrt(d))/(2*a))
    return roots


def solve_bisquare(a, b, c):
    roots = set()
    square_roots = solve_square(a, b, c)
    for root in square_roots:
        if root >= 0:
```

```
            roots.add(sqrt(root))
            roots.add(-sqrt(root))
    return roots


def main():
    a = get_coef(1, "Введите a:")
    b = get_coef(2, "Введите b:")
    c = get_coef(3, "Введите c:")
    if a == b == c == 0:
        print("Количество корней: ∞", "x ∈ ℝ", sep='\n')
        return
    roots = solve_bisquare(a, b, c)
    print("Количество корней:", len(roots))
    for root in roots:
        print(root, end=" ")


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

Текст unit_testing.py

```
import unittest
import main


class TestBisquare(unittest.TestCase):
    def test_discriminant(self):
        self.assertEqual(main.discriminant(1, 2, 1), 0)
        self.assertEqual(main.discriminant(1, 2, 3), -8)
        self.assertEqual(main.discriminant(5, 0, 5), -100)

    def test_solve_square(self):
        self.assertEqual(main.solve_square(1, 4, 3), {-3, -1})
        self.assertEqual(main.solve_square(1, 2, 1), {-1})
        self.assertEqual(main.solve_square(100, 1, 100), set())
        with self.assertRaises(ZeroDivisionError):
            main.solve_square(0, 1, 1)

    def test_solve_bisquare(self):
        self.assertEqual(main.solve_bisquare(1, 4, 3), set())
        self.assertEqual(main.solve_bisquare(1, -4, 0), {0, 2, -2})


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

Текст my_feature.feature

```
Feature: Test

  Scenario: Test my bot
    Given lab
    When discriminant calculates alright
```

```
    And solve_square gives correct roots
    And solve_bisquare gives correct roots
    Then everything is fine
```

Текст bdd_testing.py

```
from behave import *
from unit_testing import *

@given('lab')
def step_first(context):
    context.a = TestBisquare()


@when('discriminant calculates alright')
def check_discriminant(context):
    context.a.test_discriminant()


@when('solve_square gives correct roots')
def check_square(context):
    context.a.test_solve_square()


@when('solve_bisquare gives correct roots')
def check_bisquare(context):
    context.a.test_solve_bisquare()


@then('everything is fine')
def step_last(context):
    pas
```

Результат выполнения

```
✔ Tests passed: 3 of 3 tests – 1 ms
/usr/bin/python3.9 /snap/pycharm-professional/265/plugins/python/helpers/pycharm/_jb_unittest_runner.py --target unit_testing
  .TestBisquare
Testing started at 17:53 ...
Launching unittests with arguments python -m unittest unit_testing.TestBisquare in /home/levus/PycharmProjects/BKIT_2021/labs/lab4



Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```