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»
Курс «Базовые компоненты интернет-технологий»

Шаблоны проектирования и модульное тестирование в Python

Выполнил:
Студент группы ИУ5-34Б
Сергеев Никита
Дата и подпись:

Преподаватель:
Гапанюк Ю.Е.
Дата и подпись:

2021 г.

Постановка задачи

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

Файл qr.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
    '''
    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
    except:
        print(prompt, end=' ')
        coef_str = input()
    while True:
        try:
            float(coef_str)
        except ValueError:
            coef_str = input(prompt + ' ')
        else:
            break
    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
    '''
    if a==0 and b==0 and c==0:
        result = 'infinity'
    elif a==0 or b==0:
        if c > 0:
            result = list()
        elif c == 0:
            result = [0, ]
        else:
            if a == 0:
                result = [-math.sqrt(-c/b), math.sqrt(-c/b)]
            else:
                result = [-math.sqrt(math.sqrt(-c/a)), math.sqrt(math.sqrt(-c/a))]
    elif c == 0:
        if a > 0 and b > 0:
            result = [0, ]
        else:
            result = [-math.sqrt(abs(b/a)), 0, math.sqrt(abs(b/a))]
    else:
        result = list()
        D = b * b - 4 * a * c
        if D == 0.0:
            root = -b / (2.0 * a)
            result.append(root)
        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 root2 >= 0:
                result.append(-math.sqrt(root2))
                result.append(math.sqrt(root2))
    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

    # Вывод корней
    roots = get_roots(a, b, c)
    if isinstance(roots, str):
        len_roots = 1
    else:
        len_roots = len(roots)
    if len_roots == 0:
        print("Нет корней")
    elif len_roots == 1 and not isinstance(roots, str):
        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("Четыре корня: {}, {}, {} и {}".format(roots[0], roots[1], roots[2],
roots[3]))
    else:
        print("Бесконечное количество корней")


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

Файл test.py

```
import unittest
from math import sqrt
from qr import get_roots

class Tests(unittest.TestCase):
    def test1(self):
        self.assertEqual(set(get_roots(1, 2 ,3)), set())
    def test2(self):
        self.assertEqual(set(get_roots(1, -5, 6)), {-sqrt(3), -sqrt(2), sqrt(2),
sqrt(3)})
    def test3(self):
        self.assertEqual(get_roots(0, 0, 0), 'infinity')
    def test4(self):
        self.assertEqual(set(get_roots(0, 1, -4)), set([-2, 2]))
    def test5(self):
        self.assertEqual(set(get_roots(1, 0, -81)), set([-3, 3]))


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

Файл steps.py

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

@given("I have the coefficients {number1:g}, {number2:g} and {number3:g}")
def have_numbers(context, number1, number2, number3):
    context.n1 = number1
    context.n2 = number2
    context.n3 = number3

@when("I calculate them")
def calculate_roots(context):
    context.result = get_roots(context.n1, context.n2, context.n3)

@then("I expect the result to be {result}")
def expect_result(context, result):
    assert context.result == list(map(float, result.split(',')))
```

Файл test.feature

```
Feature: BDD file

    Scenario: Test1
        Given I have the coefficients 0, 1 and -4
        When I calculate them
        Then I expect the result to be -2, 2

    Scenario: Test2
        Given I have the coefficients 1, 0 and -81
        When I calculate them
        Then I expect the result to be -3, 3

    Scenario: Test3
        Given I have the coefficients 1, 2 and 0
        When I calculate them
        Then I expect the result to be 0
```

Результат выполнения

```
PS D:\лабы\1> python test.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
PS D:\лабы\1> behave test.feature
Feature: BDD file # test.feature:1

  Scenario: Test1                              # test.feature:3
    Given I have the coefficients 0, 1 and -4 # steps/steps.py:4
    When I calculate them                      # steps/steps.py:10
    Then I expect the result to be -2, 2       # steps/steps.py:14

  Scenario: Test2                               # test.feature:8
    Given I have the coefficients 1, 0 and -81 # steps/steps.py:4
    When I calculate them                       # steps/steps.py:10
    Then I expect the result to be -3, 3        # steps/steps.py:14

  Scenario: Test3                              # test.feature:13
    Given I have the coefficients 1, 2 and 0 # steps/steps.py:4
    When I calculate them                    # steps/steps.py:10
    Then I expect the result to be 0         # steps/steps.py:14

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.003s
PS D:\лабы\1>
```