



**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»
Отчет по лабораторной работе №4**

«Шаблоны проектирования и модульное тестирование в Python.»

**Выполнил:
студент группы ИУ5-33Б
Пономарева М.О.**

**Проверил:
Канев А.И.**

2021 г.

Задание:

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. 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 Моск-объектов.

Текст программы:

Файл get_roots.py

```
# -*- coding: utf-8 -*-
import math

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

```

        result.append(root1)
    root3 = (-b - sqD) / (2.0 * a)
    if root3 > 0 and math.sqrt(root3) != -root1:
        root3 = math.sqrt(root3)
        root4 = -root3
        result.append(root3)
        result.append(root4)
    if root3 == -0.0:
        root3 = 0.0
        result.append(root3)
    return result

```

Файл 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 коэффициента
    Returns:
        float: Коэффициент квадратного уравнения
    """
    try:
        # Пробуем прочитать коэффициент из командной строки
        coefStr = sys.argv[index]

        if coefStr == '0' and index == 1:
            raise

        coef = float(coefStr)
        return coef
    except:
        # Вводим с клавиатуры
        while True:
            print(prompt)
            coefStr = input()
            for i in range(len(coefStr)):
                if not(coefStr[i].isdigit() or coefStr[i] == '-'):
                    break
            else:
                break
        # Переводим строку в действительное число
        coef = float(coefStr)
        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 = []
D = b * b - 4 * a * c
if D == 0.0:
    root1 = -b / (2.0 * a)
    if root1 > 0:
        root1 = math.sqrt(root1)
        root2 = -root1
        result.append(root1)
        result.append(root2)
    if root1 == -0.0:
        root1 = 0.0
        result.append(root1)
elif D > 0.0:
    sqD = math.sqrt(D)
    root1 = (-b + sqD) / (2.0 * a)
    if root1 > 0:
        root1 = math.sqrt(root1)
        root2 = -root1
        result.append(root1)
        result.append(root2)
    if root1 == -0.0:
        root1 = 0.0
        result.append(root1)
    root3 = (-b - sqD) / (2.0 * a)
    if root3 > 0 and math.sqrt(root3) != -root1:
        root3 = math.sqrt(root3)
        root4 = -root3
        result.append(root3)
        result.append(root4)
    if root3 == -0.0:
        root3 = 0.0
        result.append(root3)
return result

def main():
    print()
    '''
    Основная функция
    '''
    a = get_coef(1, 'Введите коэффициент А:', )

    while a == 0:
        print('Коэффициент А не может быть равен 0')
        a = get_coef(1, 'Введите коэффициент А:')

    b = get_coef(2, 'Введите коэффициент В:')
    c = get_coef(3, 'Введите коэффициент С:')
    # Вычисление корней
    roots = get_roots(a, b, c)
    # Вывод корней
    len_roots = len(roots)
    if 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 {}'.format(roots[0], roots[1], roots[2]))
    elif len_roots == 4:
        print('Четыре корня: {}, {}, {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))

```

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

# Пример запуска
# gr.py 1 0 -4
```

Файл mock.py

```
from unittest import TestCase
from unittest.mock import patch
import main

class TestRoots(TestCase):
    @patch('main.get_roots', return_value=[])
    def test_no_roots(self, get_roots):
        self.assertEqual(main.get_roots(1, 2, 3), get_roots(1, 2, 3))

    @patch('main.get_roots', return_value=[0])
    def test_one_root(self, get_roots):
        expected_value = get_roots(1, 1, 0)

        cur_value = main.get_roots(1, 1, 0)
        self.assertEqual(len(cur_value), 1)
        self.assertEqual(expected_value[0] == cur_value[0], True)

    @patch('main.get_roots', return_value=[-2, 2])
    def test_two_roots(self, get_roots):
        expected_len = 2
        expected_value = get_roots(1, -2, -8)

        cur_value = main.get_roots(1, -2, -8)
        self.assertEqual(len(cur_value), expected_len)
        self.assertEqual(expected_value[0] in cur_value, True)
        self.assertEqual(expected_value[1] in cur_value, True)

    @patch('main.get_roots', return_value=[-1, 0, 1])
    def test_three_roots(self, get_roots):
        expected_len = 3
        expected_value = get_roots(1, -1, 0)

        cur_value = main.get_roots(1, -1, 0)
        self.assertEqual(len(cur_value), expected_len)
        self.assertEqual(expected_value[0] in cur_value, True)
        self.assertEqual(expected_value[1] in cur_value, True)
        self.assertEqual(expected_value[2] in cur_value, True)

    @patch('main.get_roots', return_value=[-1, 1, 0.5, -0.5])
    def test_four_roots(self, get_roots):
        expected_len = 4
        expected_value = get_roots(4, -5, 1)

        cur_value = main.get_roots(4, -5, 1)
        self.assertEqual(expected_value[0] in cur_value, True)
        self.assertEqual(expected_value[1] in cur_value, True)
        self.assertEqual(expected_value[2] in cur_value, True)
        self.assertEqual(expected_value[3] in cur_value, True)
```

Файл tdd.py

```
from unittest import TestCase
from unittest.mock import patch
import main

class TestRoots(TestCase):
    @patch('main.get_roots', return_value=[])
    def test_no_roots(self, get_roots):
        self.assertEqual(main.get_roots(1, 2, 3), get_roots(1, 2, 3))

    @patch('main.get_roots', return_value=[0])
    def test_one_root(self, get_roots):
        expected_value = get_roots(1, 1, 0)

        cur_value = main.get_roots(1, 1, 0)
        self.assertEqual(len(cur_value), 1)
        self.assertEqual(expected_value[0] == cur_value[0], True)

    @patch('main.get_roots', return_value=[-2, 2])
    def test_two_roots(self, get_roots):
        expected_len = 2
        expected_value = get_roots(1, -2, -8)

        cur_value = main.get_roots(1, -2, -8)
        self.assertEqual(len(cur_value), expected_len)
        self.assertEqual(expected_value[0] in cur_value, True)
        self.assertEqual(expected_value[1] in cur_value, True)

    @patch('main.get_roots', return_value=[-1, 0, 1])
    def test_three_roots(self, get_roots):
        expected_len = 3
        expected_value = get_roots(1, -1, 0)

        cur_value = main.get_roots(1, -1, 0)
        self.assertEqual(len(cur_value), expected_len)
        self.assertEqual(expected_value[0] in cur_value, True)
        self.assertEqual(expected_value[1] in cur_value, True)
        self.assertEqual(expected_value[2] in cur_value, True)

    @patch('main.get_roots', return_value=[-1, 1, 0.5, -0.5])
    def test_four_roots(self, get_roots):
        expected_len = 4
        expected_value = get_roots(4, -5, 1)

        cur_value = main.get_roots(4, -5, 1)
        self.assertEqual(expected_value[0] in cur_value, True)
        self.assertEqual(expected_value[1] in cur_value, True)
        self.assertEqual(expected_value[2] in cur_value, True)
        self.assertEqual(expected_value[3] in cur_value, True)
```

Папка feature, файл get_roots.feature

Feature: get_roots function

Scenario: test no roots

Given A = 1, B = 2, C = 3

When get_roots run

```
Then roots array is empty

Scenario: test one root
  Given A = 1, B = 1, C = 0
  When get_roots execute1
  Then roots is [0]

Scenario: test two root
  Given A = 1, B = -2, C = -8
  When get_roots execute2
  Then roots are [-2, 2]

Scenario: test three root
  Given A = 1, B = -1, C = 0
  When get_roots execute3
  Then roots are [-1, 0, 1]

Scenario: test four root
  Given A = 4, B = -5, C = 1
  When get_roots execute4
  Then roots are [-1, 1, 0.5, -0.5]
```

Папка feature, папка steps, файл test_no_roots.py

```
from behave import *
from get_roots import get_roots

@given('A = 1, B = 2, C = 3')
def step_impl(context):
    context.A = 1
    context.B = 2
    context.C = 3
    pass

@when('get_roots run')
def step_impl(context):
    context.array_len = len(get_roots(context.A, context.B, context.C))
    pass

@then('roots array is empty')
def step_impl(context):
    assert context.array_len is 0
```

Папка feature, папка steps, файл test_one_roots.py

```
from behave import *
from get_roots import get_roots

@given('A = 1, B = 1, C = 0')
def step_impl(context):
    context.A = 1
    context.B = 1
    context.C = 0
    pass

@when('get_roots execute1')
def step_impl(context):
    context.roots = get_roots(context.A, context.B, context.C)
    pass
```

```
@then('roots is [0]')
def step_impl(context):
    assert len(context.roots) == 1
    assert context.roots[0] == 0
```

Папка feature, папка steps, файл test two roots.py

```
from behave import *
from get_roots import get_roots

@given('A = 1, B = -2, C = -8')
def step_impl(context):
    context.A = 1
    context.B = -2
    context.C = -8
    pass

@when('get_roots execute2')
def step_impl(context):
    context.roots = get_roots(context.A, context.B, context.C)
    pass

@then('roots are [-2, 2]')
def step_impl(context):
    assert len(context.roots) == 2
    assert -2 in context.roots
    assert 2 in context.roots
```

Папка feature, папка steps, файл test three roots.py

```
from behave import *
from get_roots import get_roots

@given('A = 1, B = -1, C = 0')
def step_impl(context):
    context.A = 1
    context.B = -1
    context.C = 0
    pass

@when('get_roots execute3')
def step_impl(context):
    context.roots = get_roots(context.A, context.B, context.C)
    pass

@then('roots are [-1, 0, 1]')
def step_impl(context):
    assert len(context.roots) == 3
    assert -1 in context.roots
    assert 0 in context.roots
    assert 1 in context.roots
```

Папка feature, папка steps, файл test four roots.py

```
from get_roots import get_roots

@given('A = 4, B = -5, C = 1')
def step_impl(context):
    context.A = 4
    context.B = -5
    context.C = 1
    pass
```



```

@when('get_roots execute4')
def step_impl(context):
    context.roots = get_roots(context.A, context.B, context.C)
    pass

@then('roots are [-1, 1, 0.5, -0,5]')
def step_impl(context):
    assert len(context.roots) == 4
    assert -1 in context.roots
    assert 1 in context.roots
    assert 0.5 in context.roots
    assert -0.5 in context.roots

```

Экранные формы с примерами выполнения программы:

Результаты работы main.py:

Введите коэффициент A:

9

Введите коэффициент A:

0

Коэффициент A не может быть равен 0

Введите коэффициент A:

4

Введите коэффициент B:

10

Введите коэффициент C:

-2

Два корня: 0.43144021791495885 и -0.43144021791495885

Результаты работы tdd.py:

```

✓ Tests passed: 5 of 5 tests - 3 ms

/Users/evgeniayurova/PycharmProjects/lab4/mockimg/bin/python "/Applications/PyCharm CE.app/Contents/plugins/python-ce/help
Testing started at 01:14 ...
Launching unittests with arguments python -m unittest /Users/evgeniayurova/PycharmProjects/lab4/programm/tdd.py in /Users/

Ran 5 tests in 0.007s

OK

Process finished with exit code 0

```

Результаты работы mock.py:

```
✓ Tests passed: 5 of 5 tests – 3 ms

/Users/evgeniyurova/PycharmProjects/lab4/mocking/bin/python "/Applications/PyCharm CE.app/Contents/plugins/python-ce/help
Testing started at 01:16 ...
Launching unittests with arguments python -m unittest mock.TestRoots in /Users/evgeniyurova/PycharmProjects/Lab4/programm

Ran 5 tests in 0.007s

OK

Process finished with exit code 0
```

Результаты работы BDD тестов:

```
lab4 - programm - mock.py
Python tests for mock.TestRoots

Terminal: Local
ConfigError: No steps directory in '/Users/evgeniyurova/PycharmProjects/lab4/programm/feature/get_roots.feature'
(mocking) MacBook-Pro-Evgenia:lab4 evgeniyurova$ behave programm/features/get_roots.feature
programm/features/steps/test_no_roots.py:10: SyntaxWarning: "is" with a literal. Did you mean "=="?
assert context.array_len is 0
Feature: get_roots function # programm/features/get_roots.feature:1

Scenario: test no roots # programm/features/get_roots.feature:3
  Given A = 1, B = 2, C = 3 # programm/features/steps/test_no_roots.py:4 0.001s
  When get_roots run # programm/features/steps/test_no_roots.py:11 0.001s
  Then roots array is empty # programm/features/steps/test_no_roots.py:16 0.001s

Scenario: test one root # programm/features/get_roots.feature:8
  Given A = 1, B = 1, C = 0 # programm/features/steps/test_one_root.py:4 0.001s
  When get_roots execute1 # programm/features/steps/test_one_root.py:11 0.000s
  Then roots is [0] # programm/features/steps/test_one_root.py:16 0.000s

Scenario: test two root # programm/features/get_roots.feature:13
  Given A = 1, B = -2, C = -8 # programm/features/steps/test_two_roots.py:4 0.001s
  When get_roots execute2 # programm/features/steps/test_two_roots.py:11 0.001s
  Then roots are [-2, 2] # programm/features/steps/test_two_roots.py:16 0.001s

Scenario: test three root # programm/features/get_roots.feature:18
  Given A = 1, B = -1, C = 0 # programm/features/steps/test_three_roots.py:4 0.001s
  When get_roots execute3 # programm/features/steps/test_three_roots.py:11 0.000s
  Then roots are [-1, 0, 1] # programm/features/steps/test_three_roots.py:16 0.000s

Scenario: test four root # programm/features/get_roots.feature:23
  Given A = 4, B = -5, C = 1 # programm/features/steps/test_four_roots.py:4 0.004s
  When get_roots execute4 # programm/features/steps/test_four_roots.py:11 0.000s
  Then roots are [-1, 1, 0.5, -0.5] # programm/features/steps/test_four_roots.py:16 0.000s

1 feature passed, 0 failed, 0 skipped
5 scenarios passed, 0 failed, 0 skipped
15 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.014s

Run | TODO | Problems | Terminal | Python Packages | Python Console | Event Log
Tests passed: 5 (5 minutes ago) 11:16 LF UTF-8 4 spaces Python 3.9 (Lab4)
```