

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)

**Отчет по лабораторной работе № 4 по курсу  
Базовые компоненты интернет-технологий  
«Шаблоны проектирования и модульное тестирование в  
Python»**

ПРЕПОДАВАТЕЛЬ:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

\_\_\_\_\_  
(подпись)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-34Б

Верин Д.С.

\_\_\_\_\_  
(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 testTDD.py

```
import unittest
from qr import is_real
from qr import get_roots

class TestQr(unittest.TestCase):

    def test1(self):
        self.assertTrue(len(get_roots(0, 0, 0)) > 4)
    def test2(self):
        self.assertEqual(get_roots(1, 1, 1), [])
    def test3(self):
        self.assertEqual(get_roots(-1, -1, -1), [])
    def test4(self):
        self.assertEqual(get_roots(0, 0, 1), [])
    def test5(self):
        self.assertEqual(get_roots(9, 0, 0), [0, ])
    def test6(self):
        self.assertEqual(get_roots(0, 8, 0), [0, ])
    def test7(self):
        self.assertEqual(get_roots(31, 0, 87), [])
    def test8(self):
        self.assertEqual(get_roots(0, 1, 1), [])
    def test9(self):
        self.assertEqual(get_roots(10, 25, 0), [0, ])
    def test10(self):
        self.assertEqual(set(get_roots(10, -15, 0)), set([-6**0.5/2, 0,
6**0.5/2]))
    def test11(self):
        self.assertEqual(set(get_roots(0, 1, -16)), set([-4, 4]))
    def test12(self):
        self.assertEqual(set(get_roots(1, 0, -4)), set([-2**0.5, 2**0.5]))
    def test13(self):
        self.assertEqual(set(get_roots(1, -5, -36)), set([-3, 3]))
    def test14(self):
        self.assertEqual(set(get_roots(1, 14, 48)), set([]))
    def test15(self):
        self.assertEqual(set(get_roots(1, 1, -20)), set([-2, 2]))
    def test16(self):
        self.assertEqual(set(get_roots(1, -5, 4)), set([-2, -1, 1, 2]))
    def test17(self):
        self.assertEqual(set(get_roots(1, -5, 6)), set([-3**0.5, -2**0.5, 2**0.5,
3**0.5]))

    def test21(self):
        self.assertFalse(is_real('afaf'))
    def test22(self):
        self.assertFalse(is_real('213a'))
```

```

def test23(self):
    self.assertFalse(is_real('23a21'))
def test24(self):
    self.assertFalse(is_real('4.555555.1111'))
def test25(self):
    self.assertFalse(is_real(',01234'))
def test26(self):
    self.assertTrue(is_real('9873'))
def test27(self):
    self.assertTrue(is_real('3.9123'))
def test28(self):
    self.assertTrue(is_real(2345))
def test29(self):
    self.assertTrue(is_real(5.5))
def test30(self):
    self.assertFalse(is_real(complex(-5, 1)))

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

```

### Файл steps.py

```

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.number1 = number1
    context.number2 = number2
    context.number3 = number3

@when("I calculate them")
def calculate_roots(context):
    context.result = get_roots(context.number1, context.number2, context.number3)

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

```

### Файл testBDD.feature

Feature: My feature file using behave and  
testing my equation

Scenario: Test my biquadratic equation1

Given I have the coefficients 1, -5 and 4

When I calculate them

Then I expect the result to be -2, -1, 1, 2

Scenario: Test my biquadratic equation2

Given I have the coefficients 1, -5 and -36

When I calculate them

Then I expect the result to be -3, 3


Scenario: Test my biquadratic equation3

Given I have the coefficients 10, 25 and 0

When I calculate them

Then I expect the result to be 0

## Пример выполнения программы

 Командная строка

```
D:\python_projects\lab1>python testTDD.py
.....
-----
Ran 27 tests in 0.008s

OK

D:\python_projects\lab1>cd features

D:\python_projects\lab1\features>behave testBDD.feature
Feature: My feature file using behave and # testBDD.feature:1
  testing my equation
    Scenario: Test my biquadratic equation1      # testBDD.feature:4
      Given I have the coefficients 1, -5 and 4   # ../steps/steps.py:5
      When I calculate them                      # ../steps/steps.py:11
      Then I expect the result to be -2, -1, 1, 2 # ../steps/steps.py:15

    Scenario: Test my biquadratic equation2      # testBDD.feature:9
      Given I have the coefficients 1, -5 and -36 # ../steps/steps.py:5
      When I calculate them                      # ../steps/steps.py:11
      Then I expect the result to be -3, 3       # ../steps/steps.py:15

    Scenario: Test my biquadratic equation3      # testBDD.feature:14
      Given I have the coefficients 10, 25 and 0 # ../steps/steps.py:5
      When I calculate them                      # ../steps/steps.py:11
      Then I expect the result to be 0          # ../steps/steps.py:15

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.002s
```