**Московский государственный технический университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»
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 ИУ5-31Б
Абуховский Иван Александрович

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись и дата:

Подпись и дата:

Москва, 2021 г.

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.

Тексты программ

Файл main.py (модифицированная 1 лаб. работа)

```
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:
        # Вводим с клавиатуры
        print(prompt)
        coef_str = input()

    return coef_str


def checkingfloat(a, b, c):
    '''
    Проверка на цифру
    Args:
        a (float): коэффициент A
        b (float): коэффициент B
        c (float): коэффициент C
    Returns:
        check(boolean): результат проверки
```

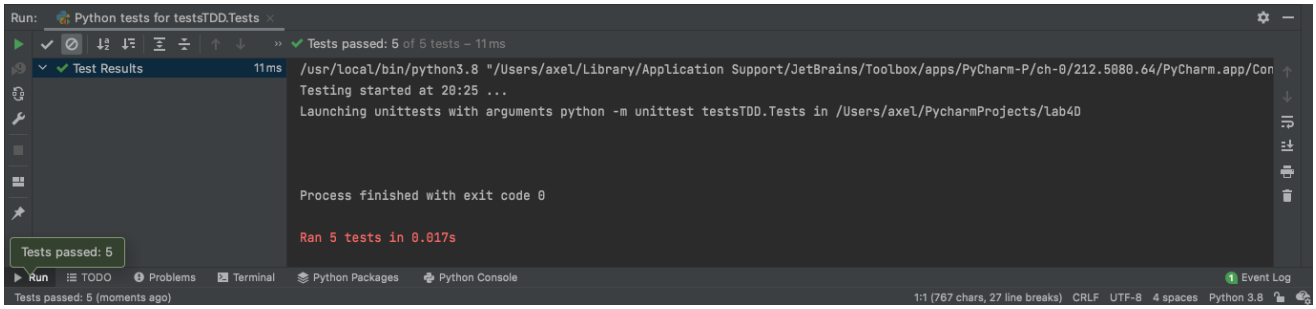
```
    '''
    check = True
    try:
        afloat = float(a)
        bfloat = float(b)
        cfloat = float(c)
    except ValueError:
        print('Ошибка!')
        check = False
    return check


def get_roots(a, b, c):
    '''
    Получение корней
    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:
        doubleroot = -b / (2.0 * a)
        if (doubleroot > 0):
            for i in range(2):
                root = (-1) ** i * math.sqrt(doubleroot)
                result.append(root)
        elif (doubleroot == 0):
            root = 0;
            result.append(root)
    elif D > 0.0:
        sqD = math.sqrt(D)
        for i in range(2):
            doubleroot = (-b + (-1) ** i * sqD) / (2.0 * a)
            if (doubleroot > 0):
                for j in range(2):
                    root = (-1) ** j * math.sqrt(doubleroot)
                    result.append(root)
            elif (doubleroot == 0):
                root = 0;
                result.append(root)

    return result


def main():
    while True:
        while True:
            a = get_coef(1, 'Введите коэффициент A:')
            b = get_coef(2, 'Введите коэффициент B:')
            c = get_coef(3, 'Введите коэффициент C:')
            if (a == '0') and (b == '0') and ((c != '0') or (c == '0')):
                print('Ошибка!')
            else:
                break
        if (checkingfloat(a, b, c)):
            break
    a = float(a)
    b = float(b)
    c = float(c)
    if (a == 0) and (c == 0):
```

```
        print('Один корень: 0')
        sys.exit()

    # 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 {} 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]))


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

(Скриншоты см. в первом отчёте)

Файл testTDD.py

```
import main
import unittest
from unittest import mock


class Tests(unittest.TestCase):

    def test_chetire_kornya(self):
        roots = main.get_roots(4, -5, 1)
        self.assertEqual([1, -1, 0.5, -0.5], roots)

    def test_tri_kornya(self):
        roots = main.get_roots(-1, 4, 0)
        self.assertEqual([0, 2, -2], roots)

    def test_dva_kornya(self):
        roots = main.get_roots(-2, 0, 10)
        self.assertAlmostEqual(1.495, roots[0], 3)
        self.assertAlmostEqual(-1.495, roots[1], 3)

    def test_nol_korney(self):
        roots = main.get_roots(1, 2, 3)
        self.assertEqual([], roots)

    @mock.patch('main.get_roots', return_value=[322])
    def test_mock(self, get_roots):
        self.assertEqual(main.get_roots(1, 2, 3), [322])
```




Файл steps.py

```python
# -*- coding: utf-8 -*-
from main import *
from behave import given, when, then


@given(u'I have {a}*x^4 + {b}*x^2 + {c} = 0')
def step_impl(context, a: float, b: float, c: float):
    context.a = float(a)
    context.b = float(b)
    context.c = float(c)


@when(u'I solve this equation')
def step_impl(context):
    context.roots = get_roots(context.a, context.b, context.c)


@then(u'I expect to get four roots: {x1}, {x2}, {x3}, {x4}')
def step_impl(context, x1: float, x2: float, x3: float, x4: float):
    result = [float(x1), float(x2), float(x3), float(x4)]
    assert context.roots == result
```

Файл bdd.feature

```
Feature: Four roots
  Scenario: roots 4 -5 1
    Given I have 4*x^4 + -5*x^2 + 1 = 0
    When I solve this equation
    Then I expect to get four roots: 1.0, -1.0, 0.5, -0.5
```