**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: Проверил:

студент группы ИУ5-35Б

Харитонов Андрей 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.

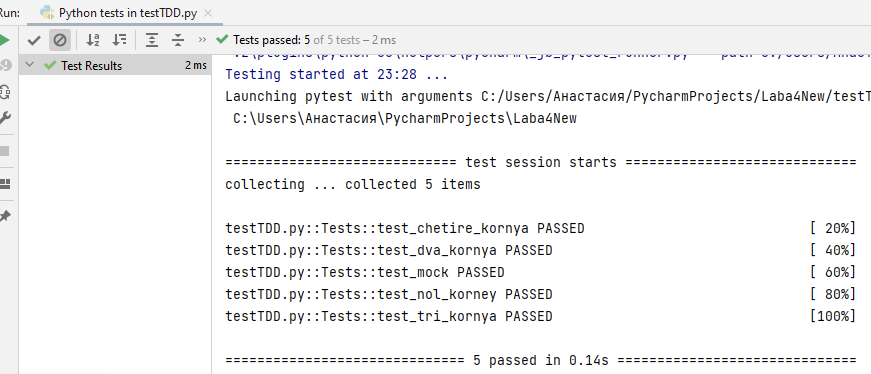
# **Тексты программ**

Файл “main.py”

import sys  
import math  
  
def get\_coef(index, prompt):try:  
 coef\_str = sys.argv[index]  
 except:print(prompt)  
 coef\_str = input()try:  
 coef = float(coef\_str)  
 except:  
 print('Значение введено неверно. Введите коэффициент занаво')  
 coef = get\_coef(index, prompt)  
 return coef  
  
  
def get\_roots(a, b, c):  
 result = []  
 while a==0:  
 print('Коэффициент а не может равняться нулю. Введите другое значение')  
 a=get\_coef(4,'Введите коэффициент а')  
 if a != 0.0 and b == 0 and c == 0:  
 return [0]  
 D = b\*b - 4\*a\*c  
 if D == 0.0:  
 root = -b / (2.0\*a)  
 if root > 0:  
 root1 = math.sqrt(root)  
 root2 = -math.sqrt(root)  
 result.append(root1)  
 result.append(root2)  
 if root == 0:  
 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:  
 root1\_ = math.sqrt(root1)  
 root2\_ = -math.sqrt(root1)  
 result.append(root1\_)  
 result.append(root2\_)  
 if root1 == 0:  
 result.append(root1)  
  
 if root2 > 0:  
 root1\_ = math.sqrt(root2)  
 root2\_ = -math.sqrt(root2)  
 result.append(root1\_)  
 result.append(root2\_)  
 if root2 == 0:  
 result.append(root2)  
  
 return result  
  
  
def output\_coef(roots):  
 len\_roots = len(roots)  
 if len\_roots == 0:  
 print('Нет корней')  
 elif len\_roots == 1:  
 print('Один корень: ', 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]))  
  
def main():  
  
 a = get\_coef(1, 'Введите коэффициент А:')  
 b = get\_coef(2, 'Введите коэффициент B:')  
 c = get\_coef(3, 'Введите коэффициент C:')roots = get\_roots(a,b,c)output\_coef(roots)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

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”

*# -\*- 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 korney: {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.feauture”

Feature: chetire kornya  
 Scenario: korni 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 korney: 1.0, -1.0, 0.5, -0.5

