**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 ИУ5-31Б преподаватель каф. ИУ5  
 Калинников Даниил Игоревич 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

Main.py – текст программы первой лабораторной работы

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 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**