**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»**

Лабораторная работа №4

по дисциплине «Базовые компоненты интернет-технологий»

Выполнила:

студентка группы ИУ5-33Б Шаповалова В.В.

Проверил: Канев А.И.

2021 г.

**Описание задания**

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:

test\_tdd

|  |
| --- |
| from main import get\_roots anw1 = set([0]) anw2 = set([-1, 0, 1]) anw3 = set([-math.sqrt(3), -math.sqrt(2), math.sqrt(2), math.sqrt(3)])  class TestGetRoots(unittest.TestCase):  def test\_area(self):  self.assertEqual(set(get\_roots(1, 0, 0)), anw1)  def test\_area(self):  self.assertEqual(set(get\_roots(1, -1, 0)), anw2)  def test\_area(self):  self.assertEqual(set(get\_roots(1, -5, 6)), anw3) |

Результат

Изображение выглядит как текст

Автоматически созданное описание

test\_bdd

|  |
| --- |
| from behave import \* from main import \*   @given("Coefs") def get\_coef(context):  context.c1 = 1  context.c2 = -4  context.c3 = 0  @when("Solved") def solution(context):  context.result = get\_roots(context.c1, context.c2, context.c3)  @then("Check result") def expect\_result(context):  assert context.result == [2, -2, 0] |
| from behave import \* from main import \*   @given("Coefs") def get\_coef(context):  context.c1 = 1  context.c2 = -4  context.c3 = 0  @when("Solved") def solution(context):  context.result = get\_roots(context.c1, context.c2, context.c3)  @then("Check result") def expect\_result(context):  assert context.result == [2, -2, 0] |

solve.feature

Feature: Test  
 Scenario: Test my function  
 Given Coefs  
 When Solved  
 Then Check result

test\_mock

|  |
| --- |
| import unittest from main import get\_roots from unittest.mock import patch, Mock  anw1 = set([0]) anw2 = set([-1, 0, 1]) class TestGetRoots(unittest.TestCase):  @patch('main.get\_roots', return\_value = [0])  def test(self, anw1):  self.assertEqual(set(get\_roots(1, 0, 0)), set([0]))   @patch('main.get\_roots', return\_value= [-1, 0, 1])  def test(self, anw2):  self.assertEqual(set(get\_roots(1, -1, 0)), set([-1, 0, 1])) |

Результат

Изображение выглядит как текст

Автоматически созданное описание