|  |  |
| --- | --- |
|  | **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 | Инженерный бизнес и менеджмент |
| КАФЕДРА | Промышленная логистика |

**Лабораторная работа №4  
«Парадигмы и конструкции языков программирования»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент |  |  |  |  |  |
|  | (группа) |  | (подпись, дата) |  | (И.О. Фамилия) |
| Преподаватель |  |  |  |  |  |
|  |  |  | (подпись, дата) |  | (И.О. Фамилия) |
|  |  |  |  |  |  |

Москва, 2024 г

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
   * TDD - фреймворк (не менее 3 тестов).
   * BDD - фреймворк (не менее 3 тестов).
   * Создание Mock-объектов (необязательное дополнительное задание).
4. from pytest\_bdd import scenario, given, when, then  
   from main import solve\_biquadratic  
     
     
   @scenario("biquadratic.feature", "Solve biquadratic equation with real roots")  
   def test\_real\_roots():  
    pass  
     
     
   @scenario("biquadratic.feature", "Solve biquadratic equation with no real roots")  
   def test\_no\_real\_roots():  
    pass  
     
     
   @scenario("biquadratic.feature", "Solve biquadratic equation with complex roots")  
   def test\_complex\_roots():  
    pass  
     
     
   @given("coefficients A, B, and C are 1, -5, and 4")  
   def coefficients\_real():  
    return 1, -5, 4  
     
     
   @given("coefficients A, B, and C are 1, 1, and 1")  
   def coefficients\_no\_real():  
    return 1, 1, 1  
     
     
   @given("coefficients A, B, and C are 1, 0, and -1")  
   def coefficients\_complex():  
    return 1, 0, -1  
     
     
   @when("I solve the equation")  
   def solve\_equation(coefficients\_real, coefficients\_no\_real, coefficients\_complex):  
    return solve\_biquadratic(\*coefficients\_real)  
     
     
   @then("I should get roots ±1 and ±2")  
   def check\_real\_roots(result):  
    assert sorted(result) == sorted([1.0, -1.0, 2.0, -2.0])  
     
     
   @then("I should get no real roots")  
   def check\_no\_real\_roots(result):  
    assert result == []  
     
     
   @then("I should get complex roots")  
   def check\_complex\_roots(result):  
    assert 1.0 in result  
    assert -1.0 in result  
     
     
   def solve\_biquadratic():  
    return None

TDD

import unittest  
from main import solve\_biquadratic  
  
  
class TestBiquadraticSolver(unittest.TestCase):  
 def test\_real\_roots(self):  
 # Уравнение x^4 - 5x^2 + 4 = 0 имеет корни: ±1, ±2  
 self.assertEqual(solve\_biquadratic(1, -5, 4), [2.0, -2.0, 1.0, -1.0])  
  
 def test\_no\_real\_roots(self):  
 # Уравнение x^4 + x^2 + 1 = 0 не имеет действительных корней  
 self.assertEqual(solve\_biquadratic(1, 1, 1), [])  
  
 def test\_complex\_roots(self):  
 # Проверяем случай, когда есть комплексные корни  
 roots = solve\_biquadratic(1, 0, -1)  
 self.assertIn(1.0, roots) # Реальный корень  
 self.assertIn(-1.0, roots) # Реальный корень  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()