

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)

**Отчет по лабораторной работе № 4 по курсу
Базовые компоненты интернет-технологий**

“Шаблоны проектирования и модульное тестирование в Python”

ПРЕПОДАВАТЕЛЬ

Гапанюк Ю. Е.

(подпись)

ИСПОЛНИТЕЛЬ:

студентка группы ИУ5-
35Б

Гурова М.Д.

(подпись)

" " _____ 2021 г.

Задание:

Задание:

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:

Для всех тестов использован одинаковый main. В случае для TDD файл называется main.py, BDD - bdd.main.py, Mock - mockmain.py.

```
# -*- coding: utf-8 -*-
```

```
import sys
```

```
import math
```

```
class Solver:
```

```
    def __init__(self):
```

```
        self.rez = []
```

```
        self.koefs = [0, 0, 0]
```

```
@property
```

```
def koef(self):
```

```
    return self.koefs
```

```
@koef.setter
```

```
def koef(self, coefs):
```

```
    self.koefs = coefs
```

```
@property
```

```
def result(self):
```

```
    return self.rez
```

```

def putter(self):
    a = self.adder(1, 'Введите коэффициент A:')
    while a == 0:
        a = self.adder(1, 'Введите коэффициент A:')
    b = self.adder(2, 'Введите коэффициент B:')
    c = self.adder(3, 'Введите коэффициент C:')
    self.koef = [a, b, c]

def adder(self, index, prompt):
    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
        float(coef_str)
    except:
        flag = True
        while flag:
            print(prompt)
            coef_str = str(input())
            if coef_str.isdigit() or (coef_str[0] == '-' and
coef_str[1:].isdigit()):
                flag = False
        return float(coef_str)

def getter(self):
    result = set()
    a, b, c = self.koef
    D = b*b - 4*a*c
    if D == 0.0:
        root = -b / (2.0 * a)
        if root > 0:
            result.add(math.sqrt(root))
            result.add(-math.sqrt(root))
        elif root == 0:
            result.add(abs(math.sqrt(root)))

    elif D > 0.0:
        root1, root2, root3, root4 = None, None, None, None
        sqD = math.sqrt(D)
        rootSq1 = (-b + sqD) / (2.0 * a)
        rootSq2 = (-b - sqD) / (2.0 * a)

```

```

        if rootSq1 > 0:
            root1 = math.sqrt(rootSq1)
            root2 = -math.sqrt(rootSq1)
        elif rootSq1 == 0:
            root1 = abs(math.sqrt(rootSq1))
        if rootSq2 > 0:
            root3 = math.sqrt(rootSq2)
            root4 = -math.sqrt(rootSq2)
        elif rootSq2 == 0:
            root3 = abs(math.sqrt(rootSq2))
        result.add(root1)
        result.add(root2)
        result.add(root3)
        result.add(root4)

    self.rez = list(filter(lambda x: x is not None, result))
    return self.rez

def printer(self):
    if len(self.rez) == 0:
        print("Нет корней")
    elif len(self.rez) == 1:
        print("Один корень: {}".format(self.rez[0]))
    elif len(self.rez) == 2:
        print("Два корня: {} и {}".format(self.rez[0],
self.rez[1]))
    elif len(self.rez) == 3:
        print("Три корня: {}, {} и {}".format(self.rez[0],
self.rez[1], self.rez[2]))
    else:
        print("Четыре корня: {}, {}, {} и
{}".format(self.rez[0], self.rez[1], self.rez[2], self.rez[3]))

def main():
    solve = Solver()
    solve.putter()
    solve.getter()
    solve.printer()

```

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

Файл TDD тестирования tester.py

```
# -*- coding: utf-8 -*-
```

```
import math
```

```
import unittest
```

```
from main import Solver
```

```
class Testing(unittest.TestCase):
```

```
    def setUp(self):
```

```
        self.solver = Solver()
```

```
    def test_koef1(self):
```

```
        self.solver.koef = [1, 1, -1]
```

```
        self.assertEqual(self.solver.koef, [1.0, 1.0, -1.0])
```

```
    def test_koef2(self):
```

```
        self.solver.koef = [0, 0, 0]
```

```
        self.assertEqual(self.solver.koef, [0, 0, 0])
```

```
    def test_result(self):
```

```
        self.solver.koef = [1, 1, -20]
```

```
        self.solver.getter()
```

```
        self.assertEqual(sorted(self.solver.result), sorted([-2,  
2]))
```

```
    def test_result2(self):
```

```
        self.solver.koef = [1, -6, 5]
```

```
        self.solver.getter()
```

```
        self.assertEqual(sorted(self.solver.result), sorted([1,  
-1, math.sqrt(5), -math.sqrt(5)]))
```

```
    def test_result3(self):
```

```
        self.solver.koef = [1, 1, -1]
```

```
        self.solver.getter()
```

```

        self.assertEqual(sorted(self.solver.result),
sorted([-0.5 * math.sqrt(-2 + 2 * math.sqrt(5)), 0.5 *
math.sqrt(-2 + 2 * math.sqrt(5))]))

```

```

if __name__ == "__main__":
    unittest.main()

```

Пример работы программы программы:

```

mac@MacBook-Pro-Mac lab4 % /usr/bin/env /usr/bin/python3 /Users/mac/.vscode/extensions/ms-python.python-2021.10.1365161279/python
Files/lib/python/debugpy/launcher 58339 -- /Users/mac/Desktop/labs/lab4/tester.py
.....
-----
Ran 5 tests in 0.002s
OK
mac@MacBook-Pro-Mac lab4 %

```

Файл BDD тестирования bddtester.py

```

from bddmain import Solver
from behave import *

```

```

@step('the user enters koefs {a}, {b}, {c}')
def step_impl(context, a, b, c):
    context.solve = Solver
    context.solve.koef = list(map(int, [a, b, c]))

```

```

@step('Finding roots')
def asd_impl(context):
    context.solve = Solver
    context.solve.result = context.solve.getter(context.solve)

```

```

@step('Testing roots {r1}, {r2}')
def abc_impl(context, r1, r2):
    context.solve = Solver
    a = sorted(context.solve.result)
    b = sorted(list(map(float, [r1, r2])))

```

```

    for i in range(len(a)):
        for j in range(len(b)):

```

```
if i == j:
    assert a[i] == b[j]
```

Файл BDD тестирования test.feature

Feature: Test Solver

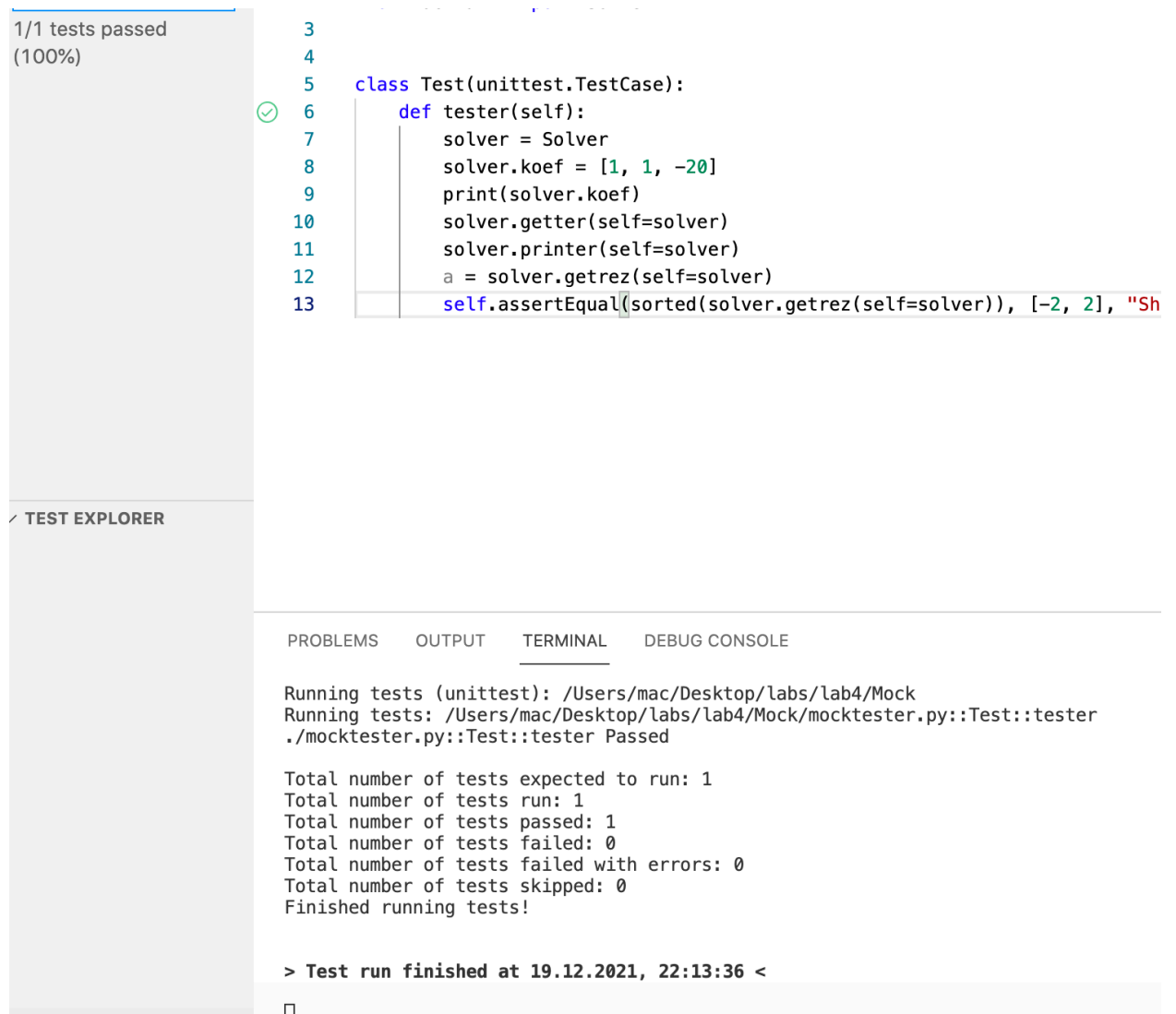
Scenario: Run test

Given the user enters coefs -1, 1, 1

When Finding roots

Then Testing roots 1, -1

Файл Mock тестирования mocktester.py



```
3
4
5 class Test(unittest.TestCase):
6     def tester(self):
7         solver = Solver
8         solver.koef = [1, 1, -20]
9         print(solver.koef)
10        solver.getter(self=solver)
11        solver.printer(self=solver)
12        a = solver.getrez(self=solver)
13        self.assertEqual(sorted(solver.getrez(self=solver)), [-2, 2], "Sh
```

TEST EXPLORER

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Running tests (unittest): /Users/mac/Desktop/labs/lab4/Mock
Running tests: /Users/mac/Desktop/labs/lab4/Mock/mocktester.py::Test::tester
./mocktester.py::Test::tester Passed

Total number of tests expected to run: 1
Total number of tests run: 1
Total number of tests passed: 1
Total number of tests failed: 0
Total number of tests failed with errors: 0
Total number of tests skipped: 0
Finished running tests!

> Test run finished at 19.12.2021, 22:13:36 <