Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №4

«Модульное тестирование»

Выполнил:

студент группы ИУ5-31Б

Сомов Кирилл

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2023 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.

Код программы

Main.py

```
from levenstein_distance import LevenshteinDistance


def main():
    word1 = input("Введите первое слово: ")
    word2 = input("Введите второе слово: ")

    distance = LevenshteinDistance.levenshtein_distance(word1, word2)
    print(f"Расстояние Левенштейна между '{word1}' и '{word2}': {distance}")


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

levenstein_distance.py

```
class LevenshteinDistance:
    @staticmethod
    def levenshtein_distance(word1, word2):
        m, n = len(word1), len(word2)
        # Создаем матрицу размером (m+1) x (n+1)
        dp = [[0] * (n + 1) for _ in range(m + 1)]

        # Инициализируем первую строку и первый столбец
        for i in range(m + 1):
            dp[i][0] = i
        for j in range(n + 1):
            dp[0][j] = j

        # Заполняем матрицу
        for i in range(1, m + 1):
            for j in range(1, n + 1):
                if word1[i - 1] != word2[j - 1]:
                    dp[i][j] = min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]) + 1
                else:
                    dp[i][j] = dp[i - 1][j - 1]
        return dp[m][n]
```

test_levenshtein_tdd.py

```
import unittest
from levenstein_distance import LevenshteinDistance


class TestLevenshteinDistance(unittest.TestCase):
    def test_same_words(self):
        self.assertEqual(LevenshteinDistance.levenshtein_distance("abc", "abc"), 0)

    def test_different_words_different_length(self):
        self.assertEqual(LevenshteinDistance.levenshtein_distance("kitten", "sitting"),
```

```
3)

    def test_different_words_same_length(self):
        self.assertEqual(LevenshteinDistance.levenshtein_distance("flaw", "lawn"), 2)

    def test_null_words(self):
        self.assertEqual(LevenshteinDistance.levenshtein_distance("", ""), 0)


if __name__ == '__main__':
    unittest.main()
```

test_levenshtein_bdd.py

```
# test_levenshtein_bdd.py
import pytest
from levenstein_distance import LevenshteinDistance


@pytest.mark.parametrize("word1, word2, expected_distance", [
    ("abc", "abd", 1),
    ("abcd", "abc", 1),
    ("abc", "abc", 0),
    ("", "", 0),
    ("kitten", "sitting", 3),
    ("flaw", "lawn", 2),
    ("developer", "develop", 2),
])
def test_levenshtein_distance(word1, word2, expected_distance):
    # When
    distance = LevenshteinDistance.levenshtein_distance(word1, word2)
    # Then
    return distance == expected_distance
```

Levenshtein_distance.feature

```
Feature: Levenshtein Distance Calculation

  Scenario Outline: Calculate Levenshtein Distance
    Given I have two words <word1> and <word2>
    When I calculate the Levenshtein distance between the words
    Then the distance should be <expected_distance>

  Examples:
    | word1       | word2       | expected_distance |
    | abc         | abc         | 0                 |
    | abc         | abcd        | 1                 |
    | abc         | abd         | 1                 |
    |             |             | 0                 |
    | kitten      | sitting     | 3                 |
    | flaw        | lawn        | 2                 |
    | developer   | develop     | 3                 |
```

Вывод

tdd

```
Testing started at 12:48 ...
Launching pytest with arguments test_levenshtein_tdd.py::TestLevenshteinDistance --no-header --no-summary -q in /Use

============================= test session starts ==============================
collecting ... collected 4 items

test_levenshtein_tdd.py::TestLevenshteinDistance::test_different_words_different_length PASSED [ 25%]
test_levenshtein_tdd.py::TestLevenshteinDistance::test_different_words_same_length PASSED [ 50%]
test_levenshtein_tdd.py::TestLevenshteinDistance::test_null_words PASSED [ 75%]
test_levenshtein_tdd.py::TestLevenshteinDistance::test_same_words PASSED [100%]

============================== 4 passed in 0.01s ===============================

Process finished with exit code 0
```

Bdd

```
============================= test session starts ==============================
collecting ... collected 7 items

test_levenshtein_bdd.py::test_levenshtein_distance[abc-abd-1] PASSED     [ 14%]
test_levenshtein_bdd.py::test_levenshtein_distance[abcd-abc-1] PASSED    [ 28%]
test_levenshtein_bdd.py::test_levenshtein_distance[abc-abc-0] PASSED     [ 42%]
test_levenshtein_bdd.py::test_levenshtein_distance[--0] PASSED           [ 57%]
test_levenshtein_bdd.py::test_levenshtein_distance[kitten-sitting-3] PASSED [ 71%]
test_levenshtein_bdd.py::test_levenshtein_distance[flaw-lawn-2] PASSED   [ 85%]
test_levenshtein_bdd.py::test_levenshtein_distance[developer-develop-2] PASSED [100%]

======================== 7 passed, 7 warnings in 0.01s =========================

Process finished with exit code 0
```