Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Факультет информационных технологий и кибербезопасности

Кафедра информационных систем и программирования

**ОТЧЕТ**

по лабораторной работе №10.

Тема работы: «Упрощение создания тестов при помощи библиотеки Fluent Assertions».

Наименование дисциплины: «Тестирование и отладка программного обеспечения».

Подготовил:

Студент группы 23-КБ-ПР1

Булгаков В. В.

Краснодар

2025

**Цель работы**

Цель работы – изучить подход к созданию тестов с помощью библиотеки Chai.

**Задание**

1) На основе класса из предыдущей лабораторной работы (вариант задания оставить прежним) создать тестирующий проект, использующий библиотеку Chai.

2) Протестировать проект.

3) Оформить отчёт.

**Ход работы**

1-3) На основе класса из предыдущей лабораторной работы создал тестирующий класс, используя библиотеку Chai, протестировал проект.

import { jest, describe, it, beforeEach, beforeAll } from '@jest/globals';

import { assert } from 'chai';

let mockGraphReaderRead;

let mockGraphWriterWrite;

jest.unstable\_mockModule('../src/GraphReader.js', () => {

  mockGraphReaderRead = jest.fn();

  return {

    default: jest.fn().mockImplementation(() => ({

      read: mockGraphReaderRead,

    })),

  };

});

jest.unstable\_mockModule('../src/GraphWriter.js', () => {

  mockGraphWriterWrite = jest.fn();

  return {

    default: jest.fn().mockImplementation(() => ({

      write: mockGraphWriterWrite,

    })),

  };

});

let Graph, GraphReader, GraphWriter, mockReader, mockWriter;

beforeAll(async () => {

  const graphModule = await import('../src/Graph.js');

  Graph = graphModule.default;

  const graphReaderModule = await import('../src/GraphReader.js');

  GraphReader = graphReaderModule.default;

  mockReader = new GraphReader();

  const graphWriterModule = await import('../src/GraphWriter.js');

  GraphWriter = graphWriterModule.default;

  mockWriter = new GraphWriter();

});

describe('ChaiGraphTest', () => {

  beforeEach(() => {

    jest.clearAllMocks();

    mockGraphReaderRead.mockReturnValue([

      [0, 1, 0, 1],

      [1, 0, 1, 0],

      [0, 1, 0, 1],

      [1, 0, 1, 0],

    ]);

    mockGraphWriterWrite.mockImplementation(() => {});

  });

  it('isConnected()', async () => {

    const adjacencyMatrix = mockReader.read();

    const incidenceMatrix = Graph.adjacencyToIncidence(adjacencyMatrix);

    const graph = new Graph(incidenceMatrix);

    assert.isTrue(graph.isConnected());

  });

  it('isNotConnected()', async () => {

    mockGraphReaderRead.mockReturnValue([

      [0, 1, 0, 0],

      [1, 0, 0, 0],

      [0, 0, 0, 1],

      [0, 0, 1, 0],

    ]);

    const adjacencyMatrix = mockReader.read();

    const incidenceMatrix = Graph.adjacencyToIncidence(adjacencyMatrix);

    const graph = new Graph(incidenceMatrix);

    assert.isFalse(graph.isConnected());

  });

  it('writeGraphTxt()', async () => {

    const adjacencyMatrix = mockReader.read();

    const incidenceMatrix = Graph.adjacencyToIncidence(adjacencyMatrix);

    const graph = new Graph(incidenceMatrix);

    const edgeList = graph.toEdgeList();

    mockWriter.write('output.txt', edgeList);

    assert.strictEqual(mockGraphWriterWrite.mock.calls.length, 1);

  });

  it('writeGraphHtml()', async () => {

    const adjacencyMatrix = mockReader.read();

    const incidenceMatrix = Graph.adjacencyToIncidence(adjacencyMatrix);

    const graph = new Graph(incidenceMatrix);

    const edgeList = graph.toEdgeList();

    mockWriter.write('output.html', edgeList);

    assert.strictEqual(mockGraphWriterWrite.mock.calls.length, 1);

  });

  it('graphReaderTxt()', async () => {

    const adjacencyMatrix = mockReader.read();

    const incidenceMatrix = Graph.adjacencyToIncidence(adjacencyMatrix);

    const graph = new Graph(incidenceMatrix);

    assert.strictEqual(mockGraphReaderRead.mock.calls.length, 1);

  });

  it('graphReaderHtml()', async () => {

    const htmlData = `

      <table>

        <tr><td>0</td><td>1</td></tr>

        <tr><td>1</td><td>0</td></tr>

      </table>

    `;

    const freshReader = new GraphReader();

    const mockRead = jest.spyOn(freshReader, 'read').mockReturnValue([[0, 1], [1, 0]]);

    const result = freshReader.read('test.html');

    const expectedMatrix = [

      [0, 1],

      [1, 0]

    ];

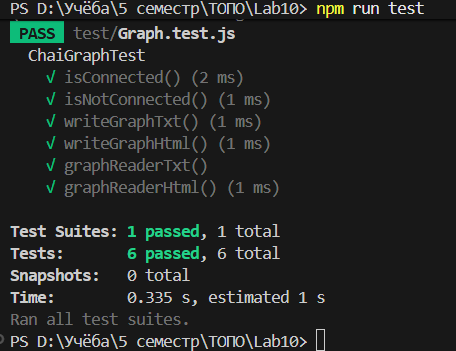
    assert.deepStrictEqual(result, expectedMatrix);

    assert.strictEqual(mockRead.mock.calls.length, 1);

    mockRead.mockRestore();

  });

});



**Вывод**

В ходе выполнения данной лабораторной работы удалось изучить подход к созданию тестов с помощью библиотеки Chai.