Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

**ОТЧЕТ**

по лабораторной работе №3.

Тема работы: «Тестирование методом черного ящика».

Наименование дисциплины: «Тестирование и отладка программного обеспечения».

Подготовил:

Студент группы 23-КБ-ПР1

Булгаков В. В.

Краснодар

2025

**Цель работы**: изучить подход к тестированию методом черного ящика.

**Задание**

1) Создать класс (в соответствии с вариантом задания из п.5), реализующий проверку принадлежности точки различным областям плоскости, задаваемых пересечением фигур.

2) Выделить классы эквивалентности.

3) Проверить граничные значения для каждого из классов эквивалентности.

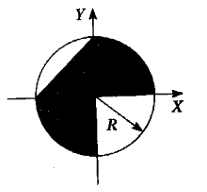
4) Составить наборы тестовых данных для созданного метода.

5) Протестировать метод на основе тестового набора с использованием программных отладочных средств.

6) Составить отчет о результатах проведенного тестирования.

Пояснение:

Создать класс, реализующий проверку принадлежности точки различным областям плоскости, задаваемых пересечением фигур. Значения, характеризующие фигуры (R, a, b, и т.д.), задать как свойства классов. В составе класса создать метод TestPoint, определяющий в какой из областей, задаваемых представленными ниже фигурами, находится точка с заданными координатами. На вход передаются координаты точки. Результатом работы метода должен быть номер области. Нумерация областей производится по усмотрению разработчика, однако должно сохраняться их соответствие между запусками программы. При разбиении на области не учитывать разделение координатными осями.



**Ход работы**:

class checkPoint {

    constructor(R) {

        if (R <= 0) {

            throw new Error("R must be > 0");

        }

        this.R = R;

    }

    #isInCircle(x, y) {

        return Math.pow(x, 2) + Math.pow(y, 2) <= Math.pow(this.R, 2)

    }

    #isInCutSector(x, y) {

        const inFourthQuadrant = x > 0 && y < 0;

        const inSecondQuadrant = x < 0 && y > 0 && y > x + this.R;

        return inFourthQuadrant || inSecondQuadrant;

    }

    TestPoint(x, y) {

        if (this.#isInCircle(x, y) && !this.#isInCutSector(x, y)) {

            return '1 область'

        } else if (this.#isInCutSector(x, y)) {

            return '2 область'

        } else {

            return '3 область'

        }

    }

}

module.exports = checkPoint;

checkpoint.test.js:

const checkPoint = require('../src/point');

describe('checkPoint', () => {

    const R = 5;

    const EPS = 0.001;

    describe('Класс 1: точка в 1 области', () => {

        test('центр круга', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(0, 0)).toBe('1 область');

            // Граничные значения

            expect(point.TestPoint(EPS, 0)).toBe('1 область');

            expect(point.TestPoint(-EPS, 0)).toBe('1 область');

            expect(point.TestPoint(0, EPS)).toBe('1 область');

            expect(point.TestPoint(0, -EPS)).toBe('1 область');

        });

        test('ниже линии y = x + R', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(-2, 2)).toBe('1 область');

            // Граничные значения

            expect(point.TestPoint(-2, 2 - EPS)).toBe('1 область');

            expect(point.TestPoint(-2, 2 + EPS)).toBe('1 область');

        });

        test('на границе круга (ось Y)', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(0, R)).toBe('1 область');

            // Граничные значения

            expect(point.TestPoint(0, R - EPS)).toBe('1 область');

            expect(point.TestPoint(0, R + EPS)).toBe('3 область'); // вне круга

        });

        test('на границе круга (ось X)', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(-R, 0)).toBe('1 область');

            // Граничные значения

            expect(point.TestPoint(-R + EPS, 0)).toBe('1 область');

            expect(point.TestPoint(-R - EPS, 0)).toBe('3 область'); // вне круга

        });

    });

    describe('Класс 2: точка в 2 области', () => {

        test('выше линии y = x + R', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(-2, 4)).toBe('2 область');

            // Граничные значения

            expect(point.TestPoint(-2, 4 - EPS)).toBe('2 область');

            expect(point.TestPoint(-2, 4 + EPS)).toBe('2 область');

        });

        test('в четвёртом квадранте', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(3, -4)).toBe('2 область');

            // Граничные значения

            expect(point.TestPoint(3 + EPS, -4)).toBe('2 область');

            expect(point.TestPoint(3 - EPS, -4)).toBe('2 область');

            expect(point.TestPoint(3, -4 + EPS)).toBe('2 область');

            expect(point.TestPoint(3, -4 - EPS)).toBe('2 область');

        });

        test('в четвёртом квадранте, на границе', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(4, -3)).toBe('2 область');

            // Граничные значения

            expect(point.TestPoint(4 + EPS, -3)).toBe('2 область');

            expect(point.TestPoint(4 - EPS, -3)).toBe('2 область');

            expect(point.TestPoint(4, -3 + EPS)).toBe('2 область');

            expect(point.TestPoint(4, -3 - EPS)).toBe('2 область');

        });

    });

    describe('Класс 3: точка в 3 области', () => {

        test('вне круга, правее', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(R + EPS, 0)).toBe('3 область');

            // Граничные значения

            expect(point.TestPoint(R, 0)).toBe('1 область'); // на границе

            expect(point.TestPoint(R - EPS, 0)).toBe('1 область'); // внутри

        });

        test('вне круга, выше', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(0, R + EPS)).toBe('3 область');

            // Граничные значения

            expect(point.TestPoint(0, R)).toBe('1 область');

            expect(point.TestPoint(0, R - EPS)).toBe('1 область');

        });

        test('вне круга, в первом квадранте', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(4, 4)).toBe('3 область');

            // Граничные значения

            expect(point.TestPoint(4 - EPS, 4 - EPS)).toBe('3 область');

            expect(point.TestPoint(4 + EPS, 4 + EPS)).toBe('3 область');

            expect(point.TestPoint(3.999, 3.999)).toBe('3 область');

        });

    });

    describe('Класс 4: точка на границе между 1 и 2', () => {

        test('на линии y = x + R, относится к 1 области', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(-1, 4)).toBe('1 область');

            // Граничные значения

            expect(point.TestPoint(-1, 4 - EPS)).toBe('1 область'); // чуть ниже

            expect(point.TestPoint(-1, 4 + EPS)).toBe('2 область'); // чуть выше

        });

        test('ещё одна точка на линии y = x + R', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(-2.5, 2.5)).toBe('1 область');

            // Граничные значения

            expect(point.TestPoint(-2.5, 2.5 - EPS)).toBe('1 область');

            expect(point.TestPoint(-2.5, 2.5 + EPS)).toBe('2 область');

        });

    });

    describe('Класс 5: точка на границе между 2 и 3', () => {

        test('на границе круга, в четвёртом квадранте', () => {

            const point = new checkPoint(R);

            expect(point.TestPoint(3, -4)).toBe('2 область');

            // Граничные значения

            expect(point.TestPoint(3 + EPS, -4)).toBe('2 область');

            expect(point.TestPoint(3 - EPS, -4)).toBe('2 область');

            expect(point.TestPoint(3, -4 + EPS)).toBe('2 область');

            expect(point.TestPoint(3, -4 - EPS)).toBe('2 область');

        });

        test('на границе круга, во втором квадранте, выше линии', () => {

            const point = new checkPoint(R);

            const x = -3.5355339059327378;

            const y = 3.5355339059327378;

            expect(point.TestPoint(x, y)).toBe('2 область');

            // Граничные значения

            expect(point.TestPoint(x + EPS, y)).toBe('2 область');

            expect(point.TestPoint(x - EPS, y)).toBe('2 область');

            expect(point.TestPoint(x, y + EPS)).toBe('2 область');

            expect(point.TestPoint(x, y - EPS)).toBe('2 область');

        });

    });

    describe('Валидация R', () => {

        test('R <= 0 вызывает ошибку', () => {

            expect(() => new checkPoint(0)).toThrow('R must be > 0');

            expect(() => new checkPoint(-5)).toThrow('R must be > 0');

        });

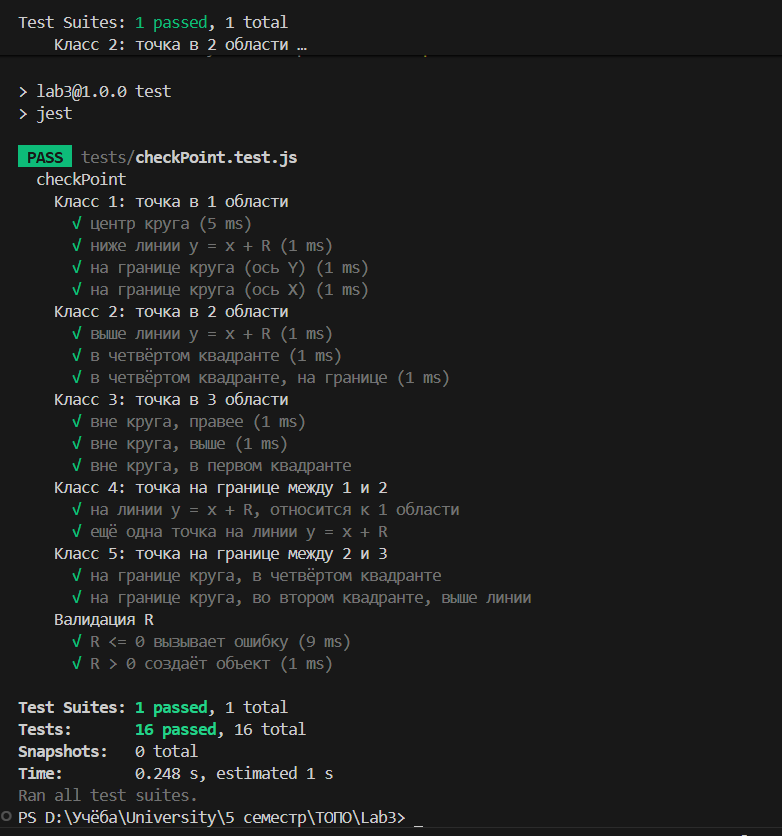
        test('R > 0 создаёт объект', () => {

            expect(() => new checkPoint(5)).not.toThrow();

        });

    });

});



Вывод: изучен подход к тестированию методом черного ящика.