Университет ИТМО

Тестирование программного обеспечения
Лабораторная работа 2

Выполнил:
Бавыкин Роман Алексеевич
Группа P33091
Вариант 9132

г. Санкт-Петербург
2024г.

1. Задание

Провести интеграционное тестирование программы, осуществляющей вычисление системы функций (в соответствии с вариантом).

Введите вариант: 9132

$$\begin{cases} \left(\left(\left((\sec(x)+\cot(x))\cdot\csc(x)\right)-\left(\frac{\tan(x)}{\csc(x)}\right)\right)\cdot\sec(x)\right)^3 & \text{if } x \le 0 \\ \left(\frac{\left(\left(\left(\log_3(x)^3\right)+\log_5(x)\right)-\log_2(x)\right)\cdot\ln(x)}{\log_{10}(x)-(\log_{10}(x)+\log_2(x))}\right) & \text{if } x > 0 \end{cases}$$

x <= 0 : (((((sec(x) + cot(x)) * csc(x)) - (tan(x) / csc(x))) * sec(x)) ^ 3)

x > 0 : (((((log_3(x) ^ 3) + log_5(x)) - log_2(x)) * ln(x)) / (log_10(x) - (log_10(x) + log_2(x))))

2. Выполнение

Цель тестирования: удостовериться, что вся система функций работает корректно при интеграции всех её модулей. Для корректности проверки общей системы будем проводить интеграционное тестирование, а все модули заменим табличными заглушками. Модули всех функций также проверим, подставляя в них заглушки, кроме базовых: sin, ln.

Методика тестирования:

Создал csv-файлы со значениями функций в определенных точках.

Для тригонометрических: от -2pi до 0 с шагом pi/6 и pi/4.

Для логарифмических: от 0 до 10 с шагом 0.5, а также e, e^2, e^3.

Разработанный тест для полной системы:

tpo2 - SystemTest.java

```java
package ru.robq;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

import java.io.IOException;

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import ru.robq.functions.Function;
import ru.robq.functions.FunctionSystem;

public class SystemTest {
    private static Function secFunction;
    private static Function cotFunction;
    private static Function cscFunction;
    private static Function tanFunction;
    private static Function lnFunction;
    private static Function log2Function;
    private static Function log3Function;
    private static Function log5Function;
    private static Function log10Function;

    private FunctionSystem system;

    @BeforeAll
    static void init() throws IOException {
        secFunction = MockUtils.createMockFromCsv(Function.class, "csv/sec.csv");
        cotFunction = MockUtils.createMockFromCsv(Function.class, "csv/cot.csv");
        cscFunction = MockUtils.createMockFromCsv(Function.class, "csv/csc.csv");
        tanFunction = MockUtils.createMockFromCsv(Function.class, "csv/tan.csv");
        lnFunction = MockUtils.createMockFromCsv(Function.class, "csv/ln.csv");
        log2Function = MockUtils.createMockFromCsv(Function.class, "csv/log2.csv");
        log3Function = MockUtils.createMockFromCsv(Function.class, "csv/log3.csv");
        log5Function = MockUtils.createMockFromCsv(Function.class, "csv/log5.csv");
        log10Function = MockUtils.createMockFromCsv(Function.class, "csv/log10.csv");
    }

    @BeforeEach
    void setUp() {
        system = new FunctionSystem(secFunction, cotFunction, cscFunction, tanFunction, lnFunction, log2Function, log3Function, log5Function, log10Function);
    }

    @Test
    void testCalculateNegativeX() {
        double result = system.calculate(-1.0472, 0.001); // -pi / 3
        assertEquals(-248.3208, result, 0.1);
    }

    @Test
    void testCalculatePositiveX() {
        double result = system.calculate(2.0, 0.001);
        assertEquals(0.22054, result, 0.1);
    }

    @Test
    void testCscZeroException() {
        assertThrows(ArithmeticException.class, () -> system.calculate(-1.0, 0.001));
    }

    @Test
    void testLogZeroException() {
        assertThrows(ArithmeticException.class, () -> system.calculate(1.0, 0.001));
    }
}

```

Тесты остальных модулей можно найти на github:
https://github.com/robqqq/tpo2

3. Вывод: ознакомился с интеграционным тестированием ПО. Научился создавать различные заглушки и работать с библиотекой Mockito.

Все классы успешно прошли модульное тестирование. Интеграционное тестирование проводилось на основе класса, решающего общую систему. В данном случае разработанное приложение строиться просто и все

разработанные модули включаются в один верхний, поэтому легко его тестировать сверху вниз.