Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине
«Тестирование программного обеспечения»

Вариант на 235726

Студент:
Группа № P33101 *Павлова А.И.*

Предподаватель: Машина.Е.А.

г.Санкт-Петербург 2024

1 Задание лабораторной работы

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

$$\begin{cases} \left(\left(\frac{(\sec(x)-\cos(x))-\tan(x)}{\sec(x)}\right)+\sec(x)\right) & \text{if } x \le 0 \\ \left(\left(\frac{\left(\left(\log_5(x)^2\right)+\ln(x)\right)\cdot\log_2(x)}{\left(\frac{\log_{10}(x)}{\log_3(x)}\right)^3}\right)\cdot\left(\log_5(x)^2\right)\right) & \text{if } x > 0 \end{cases}$$

Рис. 1: Функция для интеграционного тестирования

1. Все составляющие систему функции (как тригонометрические, так и логарифмические) должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).

2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции sin(x)):

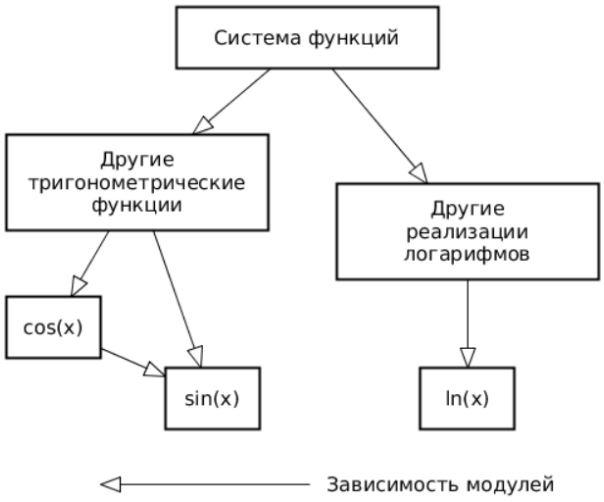

Рис. 2: Пример построения системы

3. Обе "базовые"функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.

4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.

5. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в csv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания X. Разделитель в файле csv можно использовать произвольный.

2 Процесс выполнения

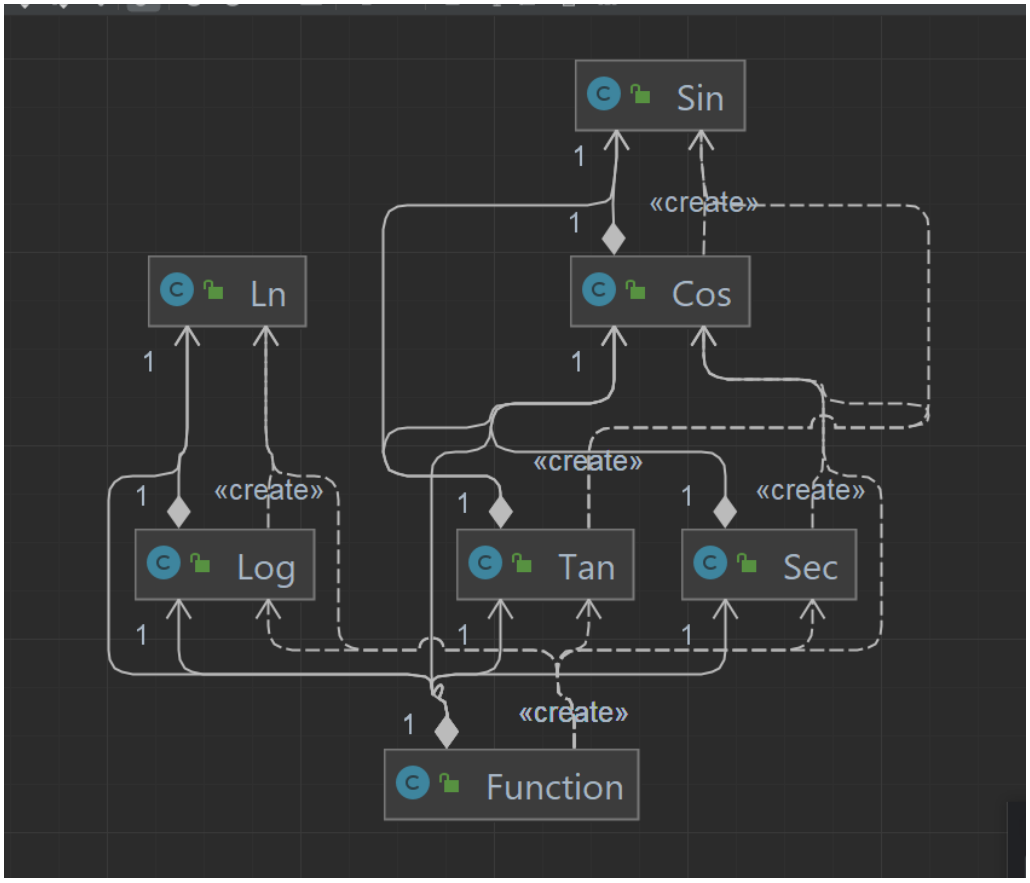

Рис. 3: Диаграмма классов

Пакеты для тетсирования были собраны так чтобы покрыть граничный знаечние типа бесконечности и Nan а также, чтобы проверить значения функции при отриацтельныхз и положительных знаечниях. Для работы использовались заглушки

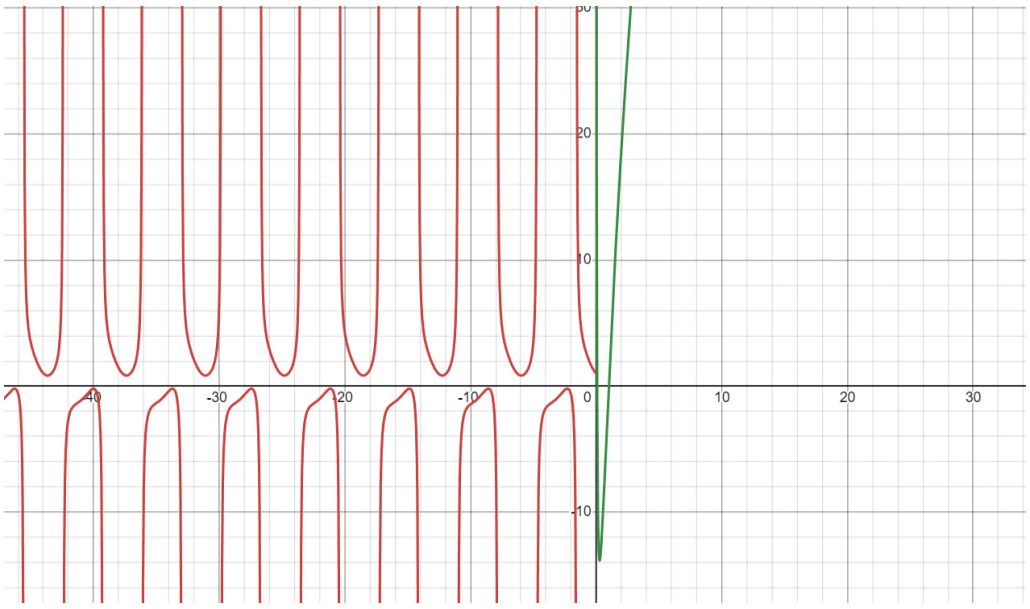

Рис. 4: График функции

```
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVRecord;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeAll;
```

```
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvFileSource;
import org.mockito.Mockito;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;

class FunctionTest {

    static double functionEps = 0.1;
    double eps = 0.1;

    static Sec secMock;
    static Cos cosMock;
    static Sin sinMock;
    static Ln lnMock;
    static Log logMock;
    static Tan tanMock;

    static Reader secIn;
    static Reader cosIn;
    static Reader sinIn;
    static Reader tanIn;
    static Reader lnIn;
    static Reader log2In;
    static Reader log3In;
    static Reader log5In;
    static Reader log10In;


    @BeforeAll
    static void init() {
        secMock = Mockito.mock(Sec.class);
        cosMock = Mockito.mock(Cos.class);
        sinMock = Mockito.mock(Sin.class);
        lnMock = Mockito.mock(Ln.class);
        logMock = Mockito.mock(Log.class);
        tanMock = Mockito.mock(Tan.class);
        try {
            secIn = new FileReader("src/main/resources/Inputs/SecIn.csv");
            cosIn = new FileReader("src/main/resources/Inputs/CosIn.csv");
            sinIn = new FileReader("src/main/resources/Inputs/SinIn.csv");
            tanIn = new FileReader("src/main/resources/Inputs/TanIn.csv");
            lnIn = new FileReader("src/main/resources/Inputs/LnIn.csv");
            log2In = new FileReader("src/main/resources/Inputs/Log2In.csv");
            log3In = new FileReader("src/main/resources/Inputs/Log3In.csv");
            log5In = new FileReader("src/main/resources/Inputs/Log5In.csv");
            log10In = new FileReader("src/main/resources/Inputs/Log10In.csv");

            Iterable<CSVRecord> records = CSVFormat.DEFAULT.parse(secIn);
            for (CSVRecord record : records) {
                Mockito.when(secMock.sec(Double.parseDouble(record.get(0)), functionEps)).thenReturn
    (Double.valueOf(record.get(1)));
            }
            records = CSVFormat.DEFAULT.parse(cosIn);
            for (CSVRecord record : records) {
                Mockito.when(cosMock.cos(Double.parseDouble(record.get(0)), functionEps)).thenReturn
    (Double.valueOf(record.get(1)));
            }
            records = CSVFormat.DEFAULT.parse(tanIn);
            for (CSVRecord record : records) {
                Mockito.when(tanMock.tan(Double.parseDouble(record.get(0)), functionEps)).thenReturn
    (Double.valueOf(record.get(1)));
            }
            records = CSVFormat.DEFAULT.parse(sinIn);
            for (CSVRecord record : records) {
                Mockito.when(sinMock.sin(Double.parseDouble(record.get(0)), functionEps)).thenReturn
    (Double.valueOf(record.get(1)));
            }
            records = CSVFormat.DEFAULT.parse(lnIn);
            for (CSVRecord record : records) {
                Mockito.when(lnMock.ln(Double.parseDouble(record.get(0)), functionEps)).thenReturn(
    Double.valueOf(record.get(1)));
            }
            records = CSVFormat.DEFAULT.parse(log2In);
            for (CSVRecord record : records) {
```

```
78                  Mockito.when(logMock.log(2, Double.parseDouble(record.get(0)), functionEps)).
    thenReturn(Double.valueOf(record.get(1)));
79              }
80              records = CSVFormat.DEFAULT.parse(log3In);
81              for (CSVRecord record : records) {
82                  Mockito.when(logMock.log(3, Double.parseDouble(record.get(0)), functionEps)).
    thenReturn(Double.valueOf(record.get(1)));
83              }
84              records = CSVFormat.DEFAULT.parse(log5In);
85              for (CSVRecord record : records) {
86                  Mockito.when(logMock.log(5, Double.parseDouble(record.get(0)), functionEps)).
    thenReturn(Double.valueOf(record.get(1)));
87              }
88              records = CSVFormat.DEFAULT.parse(log10In);
89              for (CSVRecord record : records) {
90                  Mockito.when(logMock.log(10, Double.parseDouble(record.get(0)), functionEps)).
    thenReturn(Double.valueOf(record.get(1)));
91              }
92          } catch (IOException ex) {
93              System.err.println("                                                                ");
94          }
95
96      }
97
98      @ParameterizedTest
99      @CsvFileSource(resources = "/Inputs/SystemIn.csv")
100     void testSystemWithMocks(double value, double expected) {
101         Function function = new Function(secMock,cosMock, tanMock, lnMock, logMock);
102         try {
103             Assertions.assertEquals(expected, function.writeResultToCSV(value, functionEps,
104                     new FileWriter("C:\\Users\\Anastasia\\Downloads\\tpo2\\src\\main\\resources\\
    Outputs\\SystemOut.csv", true)), eps);
105         } catch (IOException e) {
106             System.err.println("                                          ");
107         }
108
109
110     }
111
112     @ParameterizedTest
113     @CsvFileSource(resources = "/Inputs/SystemIn.csv")
114     void testWithSec(double value, double expected) {
115         Function function = new Function(new Sec(cosMock),cosMock, tanMock, lnMock, logMock);
116         Assertions.assertEquals(expected, function.SystemSolve(value, functionEps), eps);
117     }
118
119     @ParameterizedTest
120     @CsvFileSource(resources = "/Inputs/SystemIn.csv")
121     void testWithCos(double value, double expected) {
122         Function function = new Function(new Sec(new Cos(sinMock)),new Cos(sinMock), tanMock, lnMock
    , logMock);
123         Assertions.assertEquals(expected, function.SystemSolve(value, functionEps), eps);
124     }
125
126     @ParameterizedTest
127     @CsvFileSource(resources = "/Inputs/SystemIn.csv")
128     void testWithSin(double value, double expected) {
129         Function function = new Function(new Sec(new Cos(new Sin())),new Cos(new Sin()), tanMock,
    lnMock, logMock);
130         Assertions.assertEquals(expected, function.SystemSolve(value, functionEps), eps);
131     }
132
133     @ParameterizedTest
134     @CsvFileSource(resources = "/Inputs/SystemIn.csv")
135     void testWithLog(double value, double expected) {
136         Function function = new Function(secMock, cosMock, tanMock, lnMock, new Log(lnMock));
137         Assertions.assertEquals(expected, function.SystemSolve(value, functionEps), eps);
138     }
139
140     @ParameterizedTest
141     @CsvFileSource(resources = "/Inputs/SystemIn.csv")
142     void testWithLn(double value, double expected) {
143         Function function = new Function(secMock, cosMock, tanMock, new Ln(), new Log(new Ln()));
144         Assertions.assertEquals(expected, function.SystemSolve(value, functionEps), eps * 20);
145     }
146
147     @ParameterizedTest
148     @CsvFileSource(resources = "/Inputs/SystemIn.csv")
149     void testWithSinAndLn(double value, double expected) {
```

```
150         Function function = new Function();
151         Assertions.assertEquals(expected, function.SystemSolve(value, functionEps), eps * 20);
152     }
153 }
```

Листинг 1: Тестирование

3 Вывод

В данной лабораторной работе мною были освоены основы интеграцинного тестирования , исопльзование заглушек и др.