УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Распределенные системы хранения данных»

Лабораторная работа №2

Вариант 9201

Студент

Чебоксаров Я.М.

P33091

Преподаватель

Егошин А.В.

Санкт-Петербург, 2024 г.

Оглавление

Описание задания

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

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

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.

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

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.

Лабораторная работа #2

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

Введите вариант: 9201

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

x <= 0 : (((((csc(x) + cos(x)) / cos(x)) ^ 3) ^ 2) / (sin(x) * (cot(x) / csc(x))))

x > 0 : (((((log_3(x) * log_2(x)) * log_3(x)) / (ln(x) + log_10(x))) - log_10(x)) ^ 2)

Цели тестирования

Function System

1. Проверить правильно ли составлена формула
2. Проверить корректность взаимодействия с модулями, используемых функций

Cos

1. Проверить правильно ли работает функция для подсчёта косинуса на основе синуса
2. Проверить корректность интеграции с модулем Sin

Csc

1. Проверить правильно ли работает функция для подсчёта косинуса на основе синуса
2. Проверить корректность интеграции с модулем Sin

Cot

1. Проверить правильно ли работает функция для подсчёта косинуса на основе синуса
2. Проверить корректность интеграции с модулем Sin

Ln

1. Проверить правильно ли работает функция для подсчёта Ln

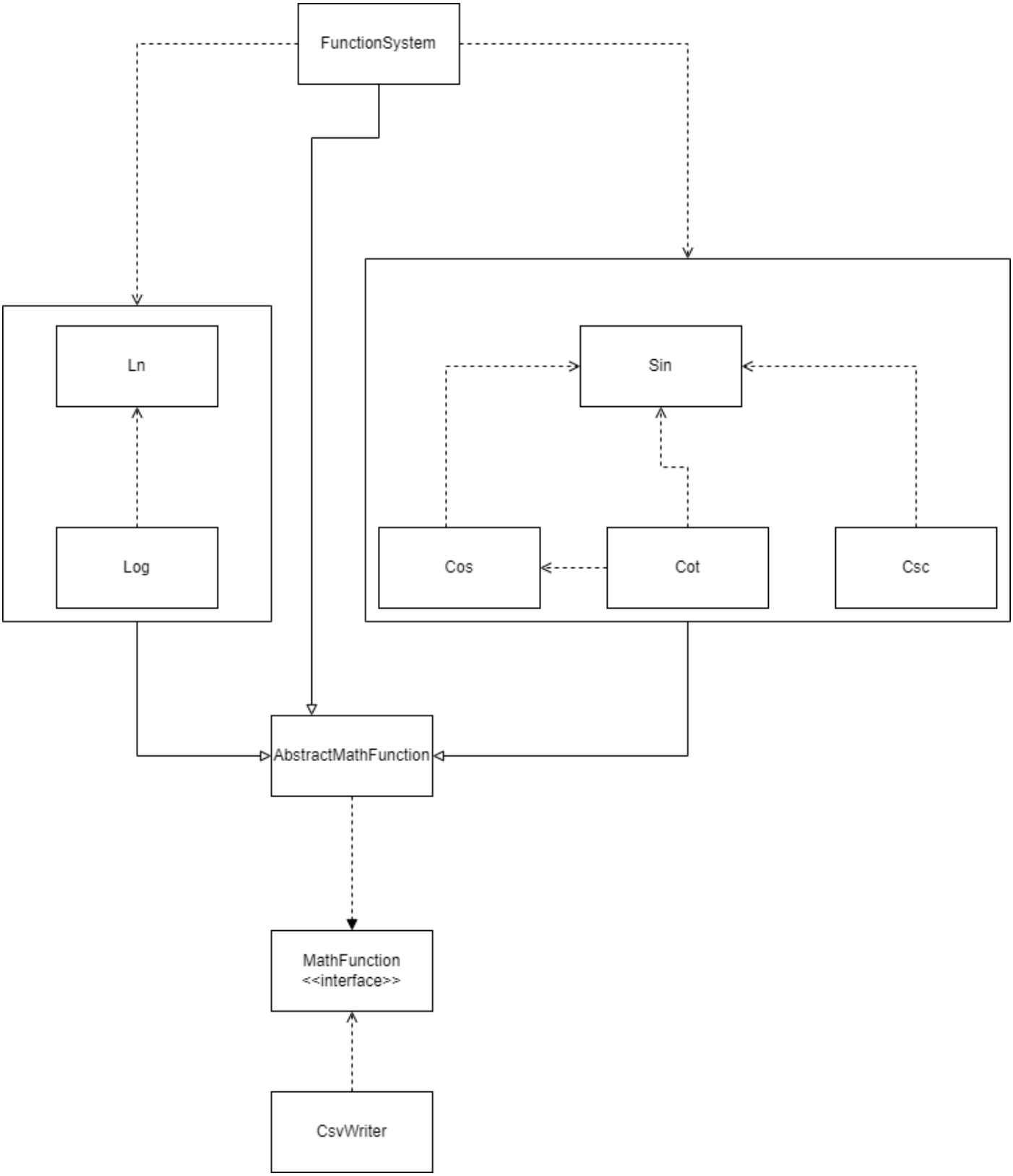
Log

1. Проверить правильно ли работает функция для подсчёта логарифма по основанию N
2. Проверить корректность интеграции с модулем Ln

Sin

1. Проверить правильно ли работает функция для синуса

Диаграмма классов


FunctionSystem
Ln
Log
Sin
Cos
Cot
Csc
AbstractMathFunction
MathFunction
<<interface>>
CsvWriter

Примеры тестов

```
    @Nested
    class CscTest{
        @ParameterizedTest
        @CsvFileSource(resources = "/csc.csv", numLinesToSkip = 1)
        public void cscFullyIsolatedTest(double val, double expectedResult,
double sinMockResult) {
            Sin sin = Mockito.mock(Sin.class);
            Mockito.when(sin.calculate(BigDecimal.valueOf(val), preci-
sion)).thenReturn(BigDecimal.valueOf(sinMockResult));
            assertEquals(expectedResult, csc.calculate(BigDecimal.val-
ueOf(val), precision).doubleValue(), tolerance);
        }

        @ParameterizedTest
        @CsvFileSource(resources = "/csc.csv", numLinesToSkip = 1)
        public void cscBasicTest(double val, double expectedResult, double
sinMockResult) {
            assertEquals(expectedResult, csc.calculate(BigDecimal.val-
ueOf(val), precision).doubleValue(), tolerance);
        }

        @ParameterizedTest
        @ValueSource(doubles = {0, 3.14159, 6.28318})
        public void cscZeroSinTest(double val) {
            assertThrows(ArithmeticException.class, () -> csc.calcu-
late(BigDecimal.valueOf(val), precision));
        }
    }
```

Ссылка на репозиторий с исходным кодом.

Функции.

Sin.

Область определения: $x \in R$

Cos.

Область определения: $x \in R$

Csc.

$$\csc(x) = \frac{1}{\sin(x)}$$

Неопределён при x кратном PI(когда sin(x) == 0)

Например: 0, Pi, 2Pi, …

Cot.

$$\cot(x) = \frac{1}{\operatorname{tg}(x)} = \frac{\cos(x)}{\sin(x)}$$

Неопределён при x кратном PI(когда sin(x) == 0)

Ln.

Область определения: $x > 0$

Log.

Область определения: $x > 0$

Формулы для wolfram:

((((csc(x) + cos(x)) / cos(x)) / (sin(x) * (cot(x) / csc(x))))^3)^2 at x=-1.1

(((((log_3(x) * log_2(x)) * log_3(x)) / (ln(x) + log_10(x))) - log_10(x)) ^ 2)

Выводы

В ходе выполнения лабораторной работы познакомился с Mokito – средством для создания заглушек, так же на практике освоил основы интеграционного тестирования, в результате сравнения данной работы с лабораторной работой номер один, понял разницу между модульным и интеграционным тестированием.