Университет ИТМО

Тестирование программного обеспечения

Лабораторная работа №2

Вариант 659

Выполнил: Саржевский Иван

Группа: P3402

г. Санкт-Петербург

2020 г.

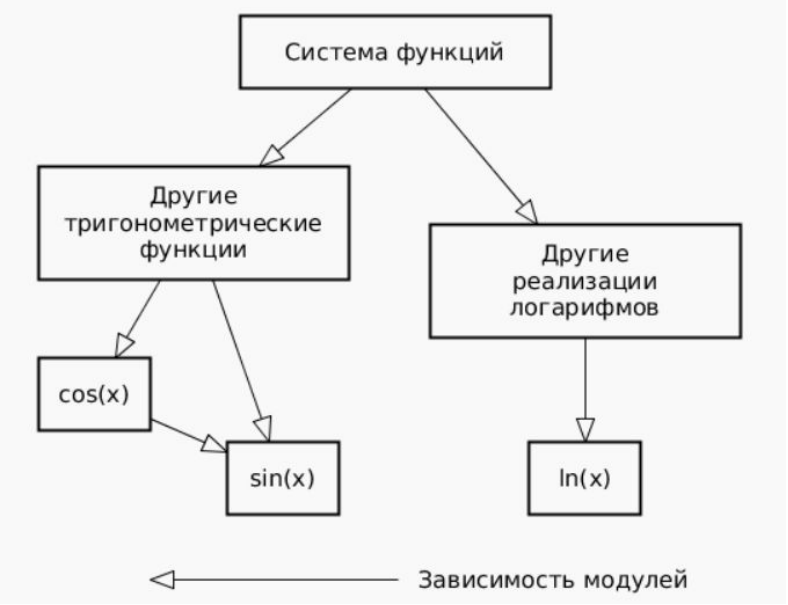
Задание

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

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

Правила выполнения работы:

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)):



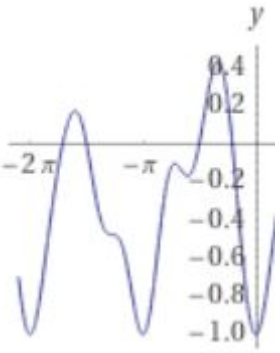
1. Обе "базовые" функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
2. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых

значений функций, и, при необходимости, определить взаимозависимые точки в модулях.

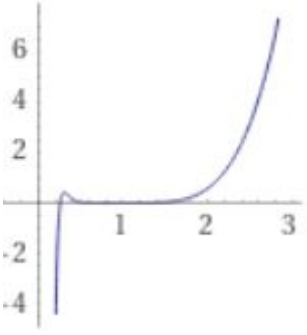
3. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в csv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания X. Разделитель в файле csv можно использовать произвольный.

Функция

При X<=0:



При X>0:



Исходный код:

https://github.com/johnny-keker/itmo-4th-year/tree/master/testing/Lab2

UML диаграмма классов:

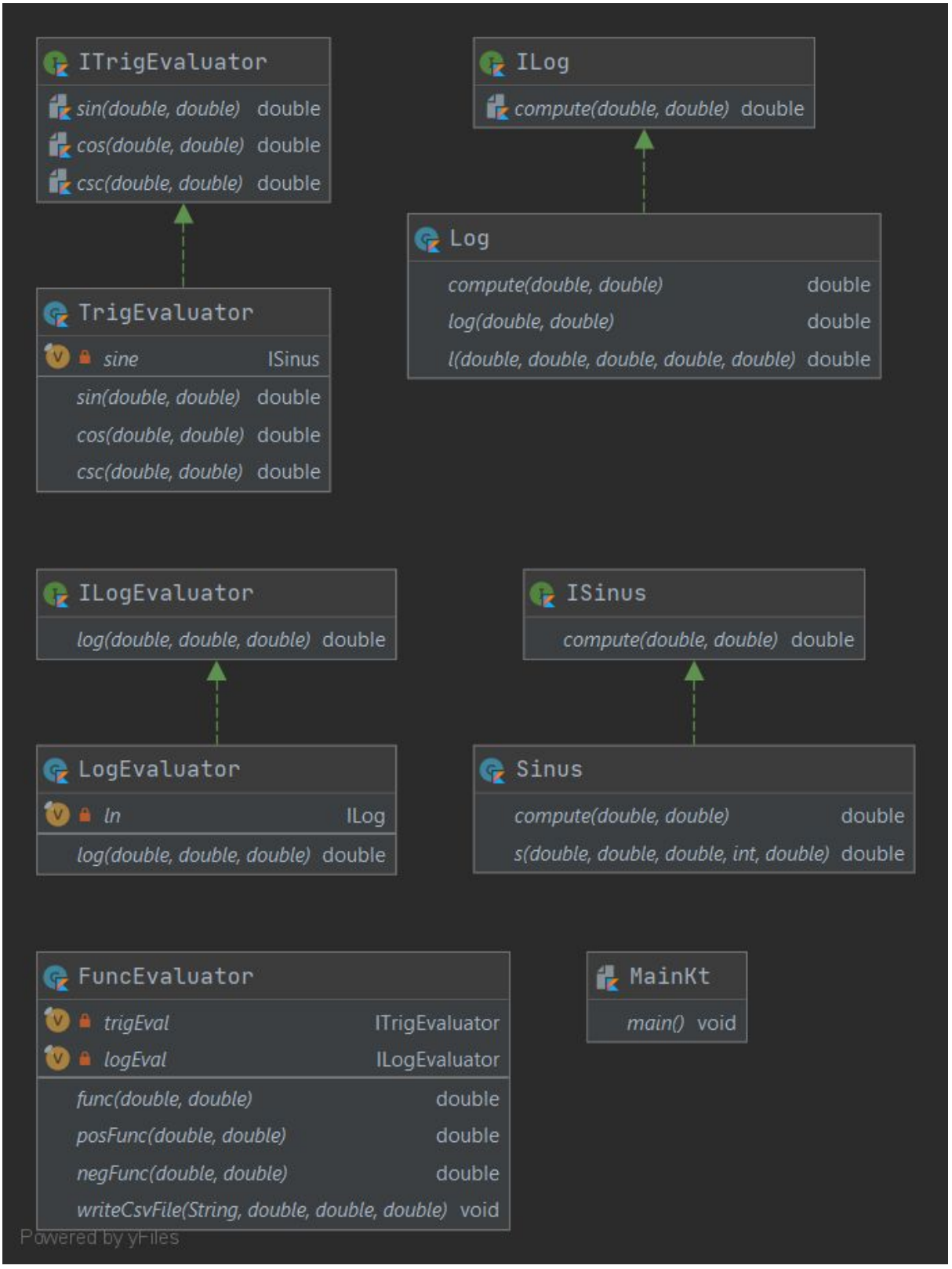
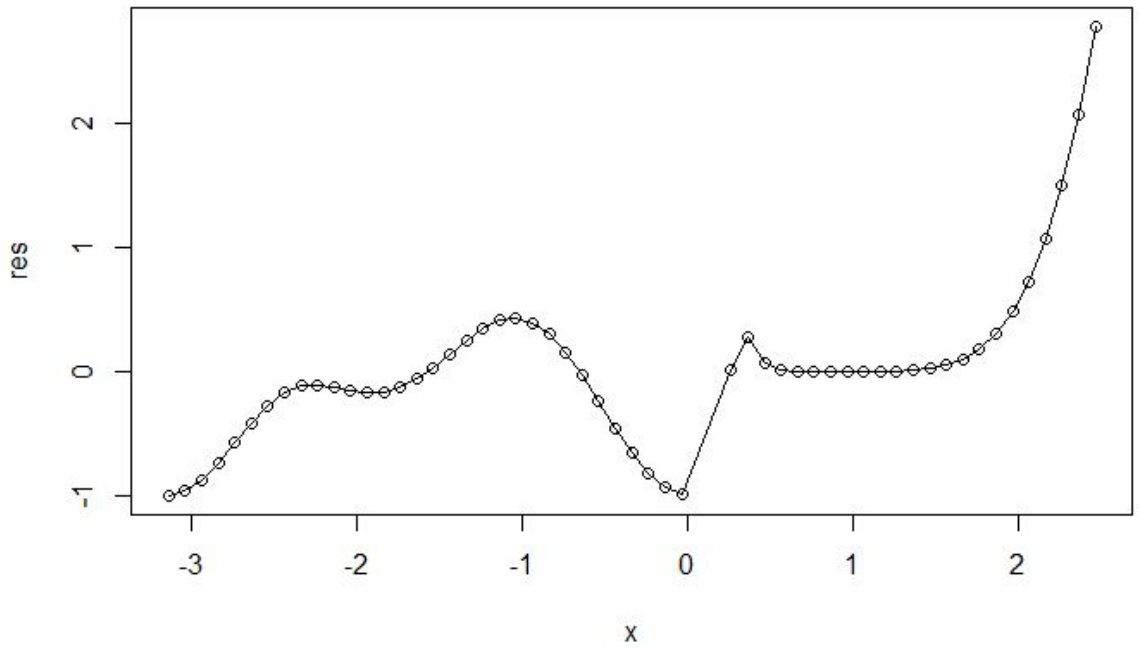


График полученной функции:



Вывод:

В ходе выполнения этой лабораторной работы были получены навыки интеграционного тестирования и была изучена библиотека Mockito.

Реализовано две базовых функции - синус (класс Sinus, интерфейс ISinus) и натуральный логарифм (класс Log, интерфейс ILog). Затем на их основе реализованы модули, которые вычисляют тригонометрические функции (класс TrigEvaluator, интерфейс ITrigEvaluator) и логарифмические функции (класс LogEvaluator и интерфейс ILogEvaluator). А уже на их основе, в классе FuncEvaluator вычисляется итоговая функция.

Можно заметить, что каждый вычислительный модуль (кроме FuncEvaluator) реализует одноименный интерфейс. Это удобно с точки зрения тестирования, так как позволяет писать тесты с использованием Mock-объектов еще до появления конкретной имплементации этих интерфейсов.

Тестирование итоговой функции производилось в несколько этапов на разных уровнях. На самом высоком уровне объекты ITrigEvaluator и ILogEvaluator заменялись на Mock-объекты и тестировался FuncEvaluator. Проверялось, что вызываются необходимые тригонометрические и логарифмические функции необходимое количество раз, а также сама арифметика подсчета итоговой функции на предопределенных значениях тригонометрических и логарифмических компонент, кроме этого проверялась обработка невалидных входных параметров. Также на уровне итоговой функции было проведено интеграционное тестирование, при котором

не было создано Mock-объектов и проверялось поведение функции на реальных реализациях всех базовых элементов. Тестирование было проведено на всех экстремумах и точках пересечения с осью OX, а также в некоторых произвольных точках.

Затем тестировались ITrigEvaluator и ILogEvaluator, с использованием Mock'ов базовых функций, sin и ln соответственно. Проверялись негативные кейсы, а также то, сколько раз и с какими аргументами базовые функции вызываются. Затем на этом уровне было проведено интеграционное тестирование с реализациями синуса и натурального логарифма. При этом производные функции тестировались аналогично синусу в первой лабораторной работе - с учетом особенностей функций и использованием классов эквивалентности.

Затем были протестированы самые базовые функции - sin и ln. Они тестировались аналогично первой лабораторной работе.

Для инициализации необходимых компонентов использовались методы, помеченные аннотацией @BeforeEach. Также, для удобства редактирования тестовых случаев и лаконичности кода использовались параметризованные тесты, источником данных для которых были csv файлы, для чего пригодились аннотации @ParameterizedTest и @CsvFileSource.

Помимо всего прочего, было разработано консольное приложение, генерирующее csv файл со значениями функции на заданном интервале с заданным шагом прироста. По данным из такого csv файла был построен график, приведенный в отчете (можно заметить, что на нем не достает несколько точек в окрестности нуля, однако это обусловлено особенностью поведения функции - при приближении к нулю со стороны положительных чисел функция быстро стремится к минус бесконечности, что делало невозможным представить все значения на одном графике).