Федеральное государственное автономное образовательное учреждение высшего образования "Национальный Исследовательский Университет ИТМО"
Мегафакультет Компьютерных Технологий и Управления
Факультет Программной Инженерии и Компьютерной Техники

Вариант №133211
Лабораторная работа 2
по дисциплине
Тестирование программного обеспечения

Выполнил Студент группы P33102
Лапин Алексей Александрович
Преподаватель:
Харитонова Анастасия Евгеньевна

г. Санкт-Петербург
2024г.

Задание:

Лабораторная работа #2

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

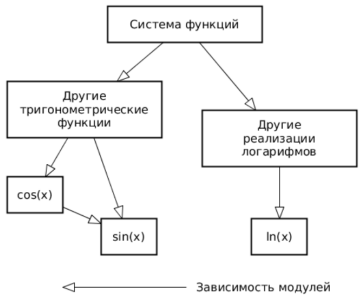
Введите вариант: 133211

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

x <= 0 : (((sec(x) ^ 3) / csc(x)) / (sin(x) + tan(x)))
x > 0 : (((((log_3(x) ^ 3) / ln(x)) / (ln(x) * log_3(x))) - log_5(x)) - log_5(x))

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)):

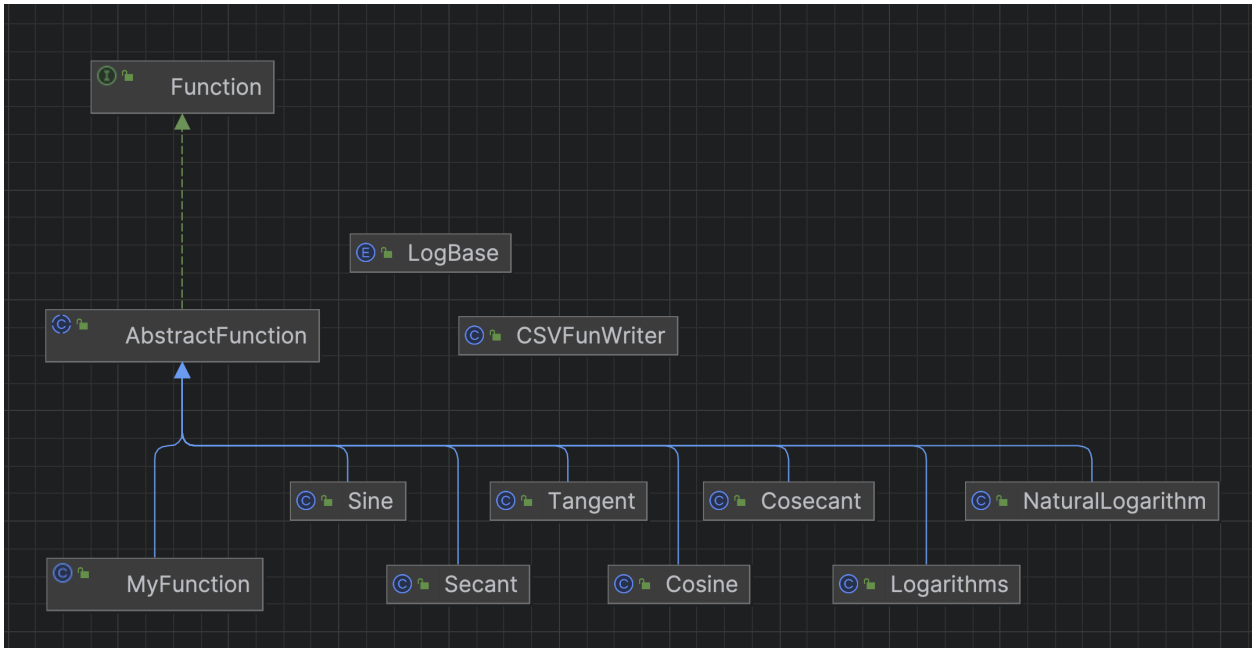


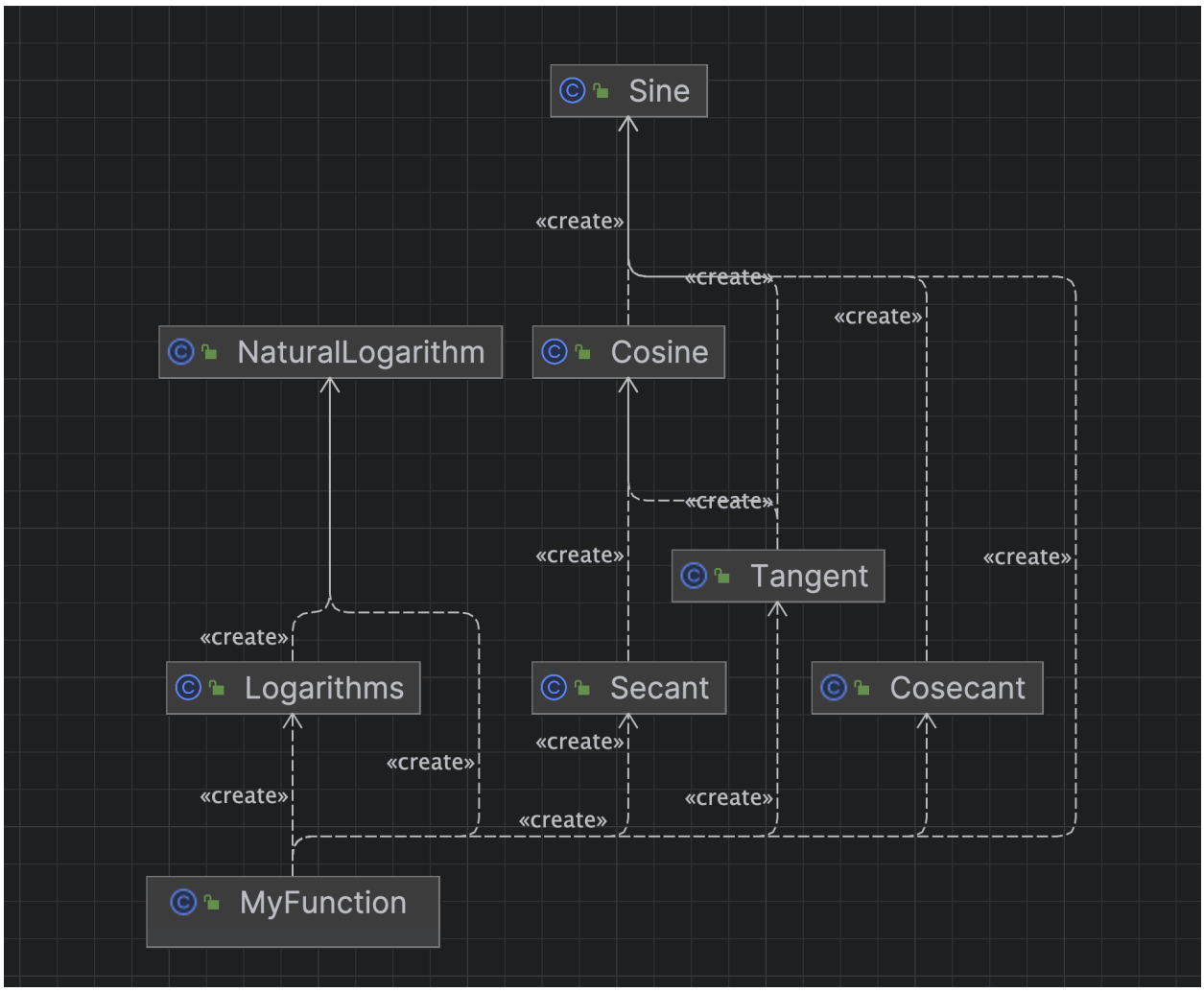
3. Обе "базовые" функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки.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

Порядок выполнения работы:

1. Разработать приложение, руководствуясь приведёнными выше правилами.
2. С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт https://www.wolframalpha.com/.
3. Собрать приложение, состоящее из заглушек. Провести интеграцию приложения по 1 модулю, с обоснованием стратегии интеграции, проведением интеграционных тестов и контролем тестового покрытия системы функций.

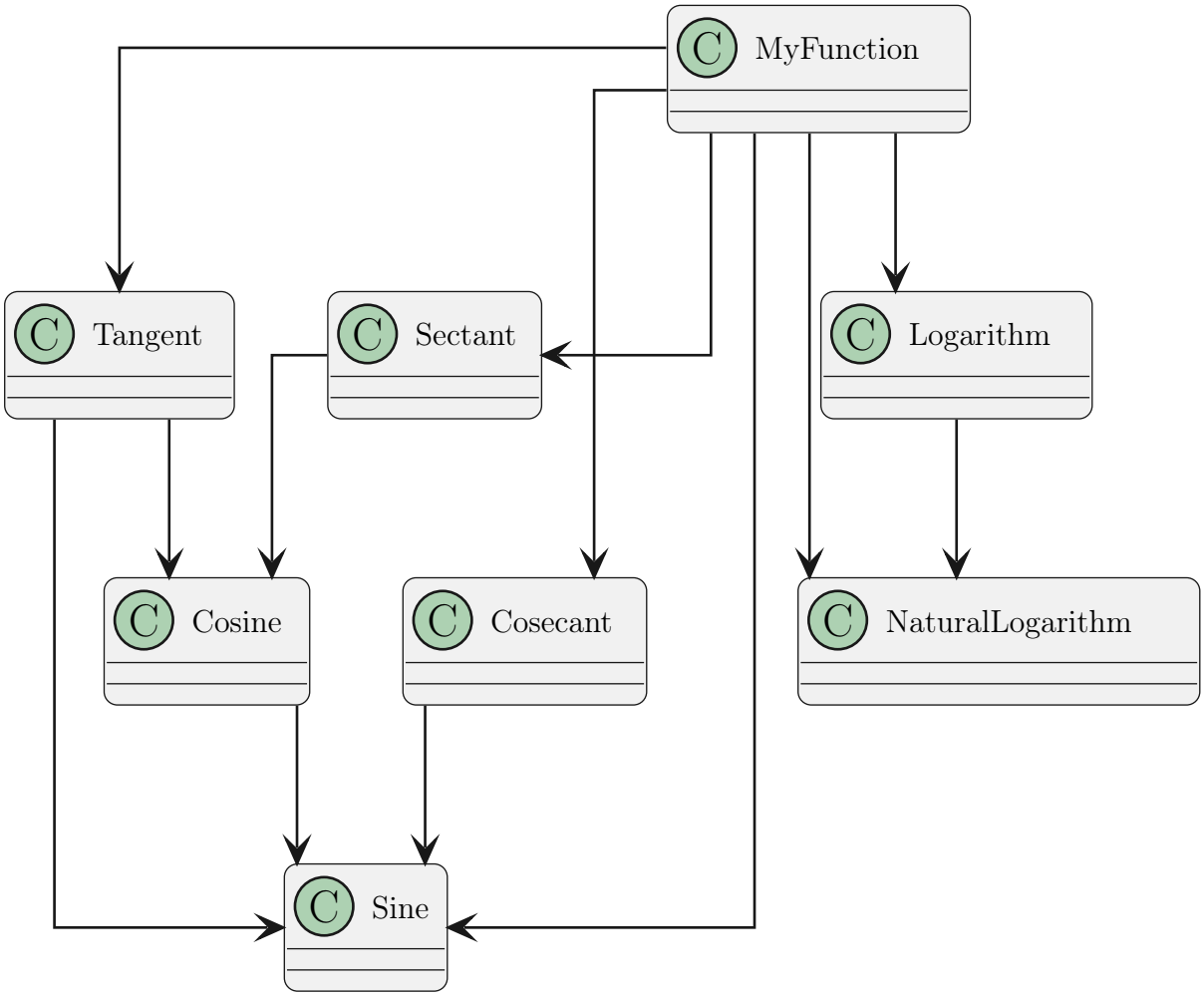
UML-диаграмма классов разработанного приложения.




Sine
«create»
«create»
«create»
NaturalLogarithm
Cosine
«create»
«create»
«create»
Tangent
«create»
«create»
Logarithms
Secant
Cosecant
«create»
«create»
«create»
«create»
«create»
MyFunction

Описание тестового покрытия с обоснованием его выбора.

Test coverage diagram



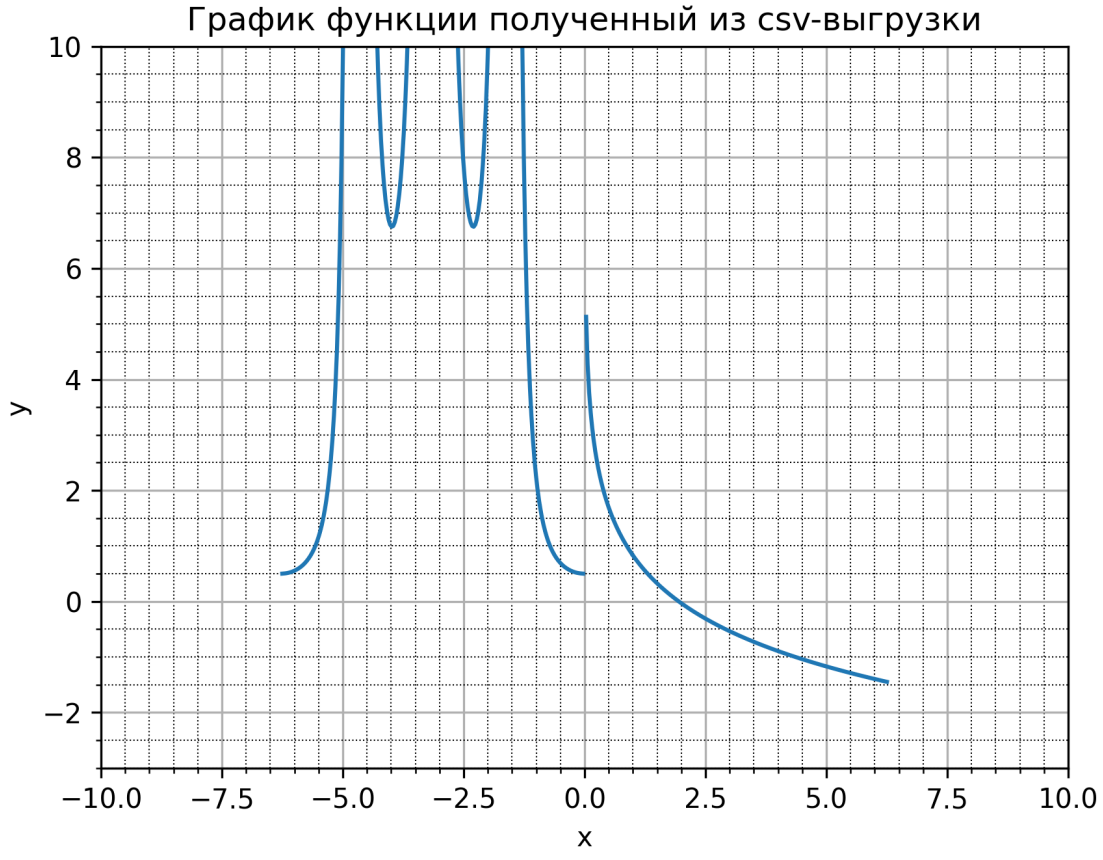
Интеграция приложения проводилась сверху вниз, так как этот метод позволяет использовать заглушки, которые гораздо проще реализовать и в целом данный вид интеграции более прост в реализации. Тестовое покрытие состоит из 2 частей:

1. Модульные тесты для каждого класса.
2. Интеграционные тесты для проверки взаимодействия классов в MyFunction.

Coverage AllTests

Element	Class, %	Method, %	Line, %	Branch, %
ru.ifmo	100% (11/11)	100% (27/27)	100% (118/118)	100% (54/54)
logarithms	100% (3/3)	100% (7/7)	100% (30/30)	100% (16/16)
Logarithms	100% (1/1)	100% (3/3)	100% (7/7)	100% (0/0)
LogBase	100% (1/1)	100% (3/3)	100% (6/6)	100% (0/0)
NaturalLogarithm	100% (1/1)	100% (1/1)	100% (17/17)	100% (16/16)
trigonometry	100% (5/5)	100% (13/13)	100% (54/54)	100% (28/28)
Cosecant	100% (1/1)	100% (3/3)	100% (8/8)	100% (4/4)
Cosine	100% (1/1)	100% (3/3)	100% (8/8)	100% (2/2)
Secant	100% (1/1)	100% (3/3)	100% (8/8)	100% (4/4)
Sine	100% (1/1)	100% (1/1)	100% (19/19)	100% (14/14)
Tangent	100% (1/1)	100% (3/3)	100% (11/11)	100% (4/4)
AbstractFunction	100% (1/1)	100% (2/2)	100% (3/3)	100% (0/0)
CSVFunWriter	100% (1/1)	100% (2/2)	100% (6/6)	100% (2/2)
Function	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
MyFunction	100% (1/1)	100% (3/3)	100% (25/25)	100% (8/8)

Графики, построенные csv-выгрузкам, полученным в процессе интеграции приложения.



Отличие только в количестве точек.

Графики других функций:

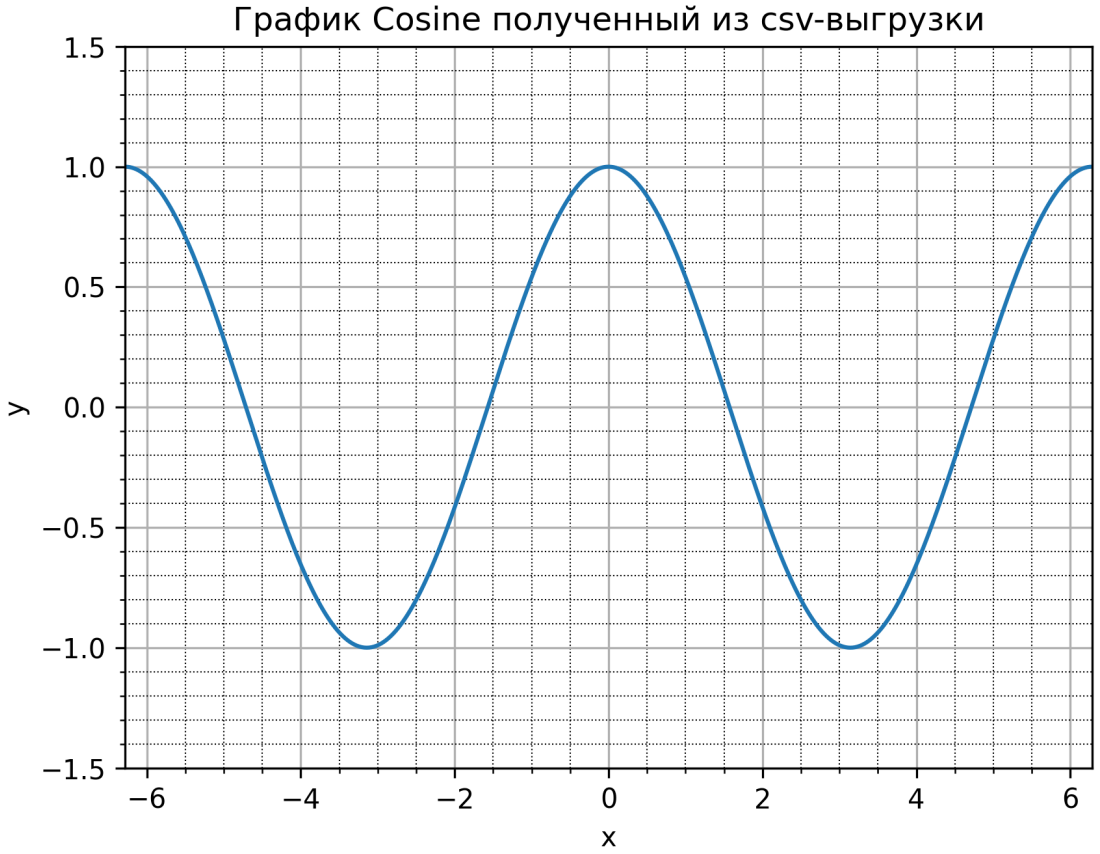


График Cosine полученный из csv-выгрузки
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−6
−4
−2
0
2
4
6
x

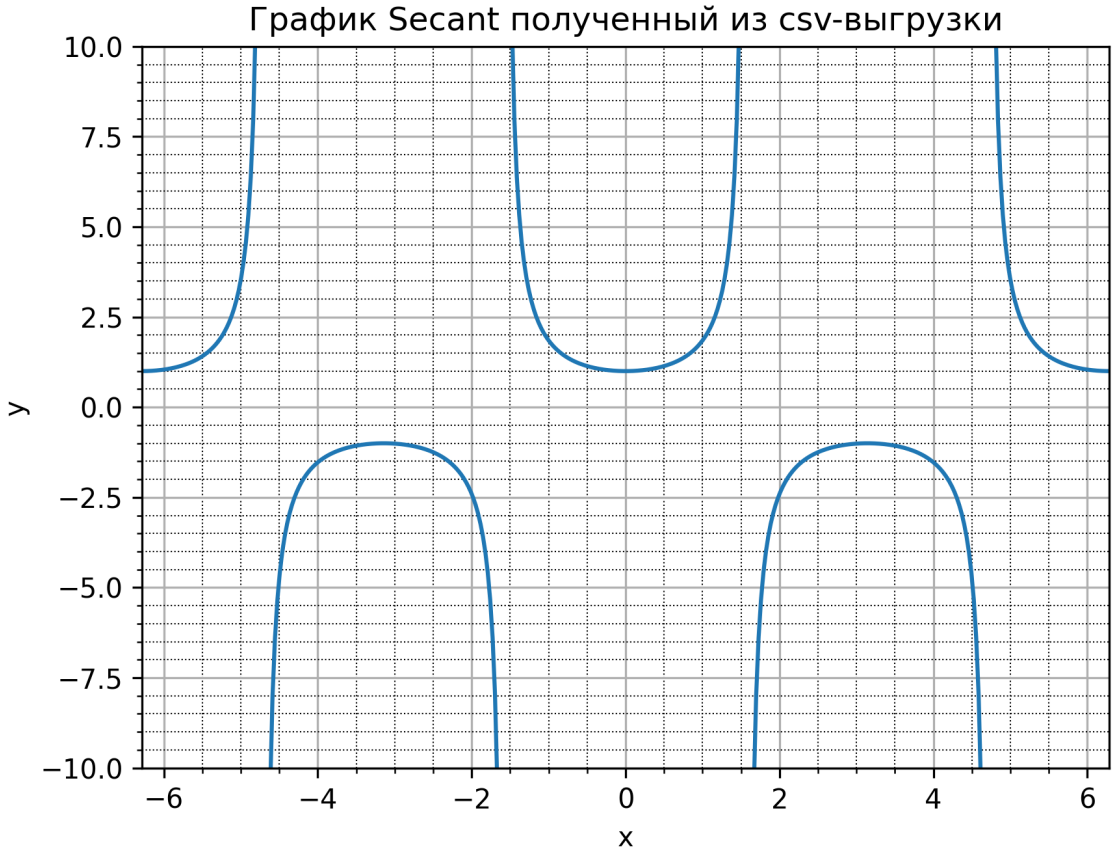


График Secant полученный из csv-выгрузки
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
y
−6
−4
−2
0
2
4
6
x

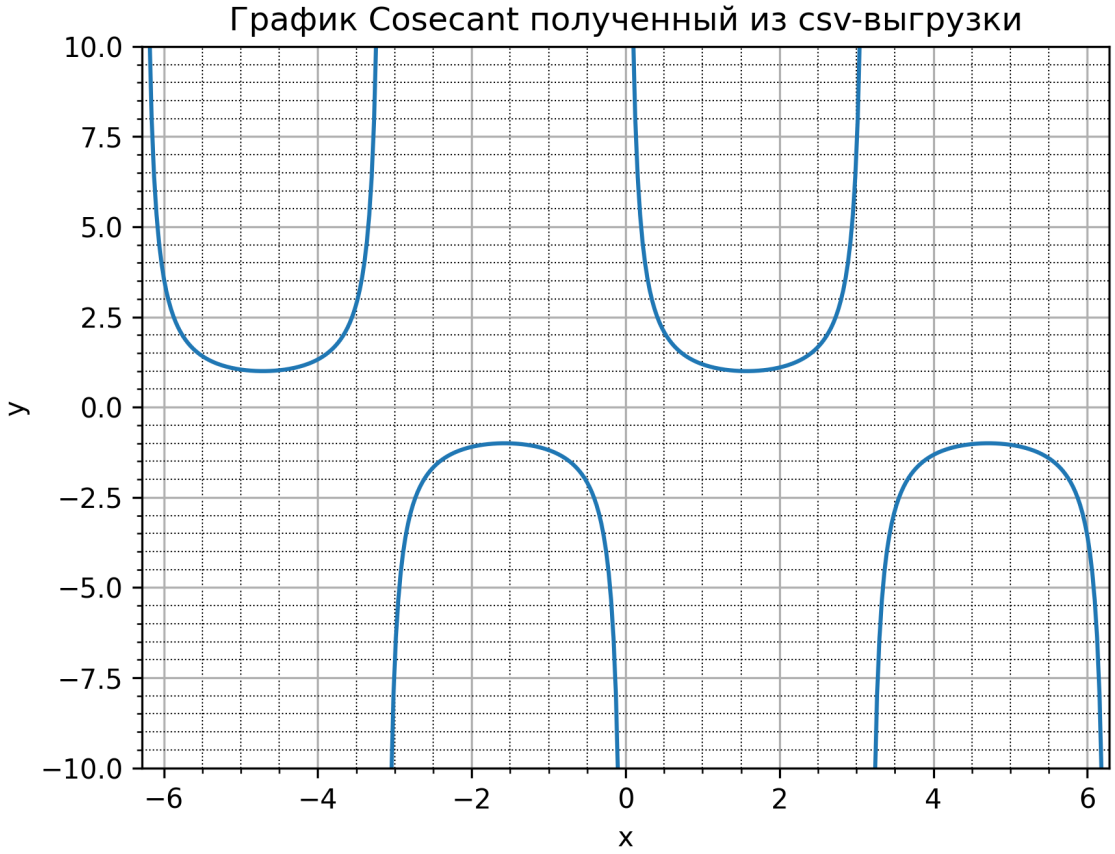


График Cosecant полученный из csv-выгрузки
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
y
−6
−4
−2
0
2
4
6
x

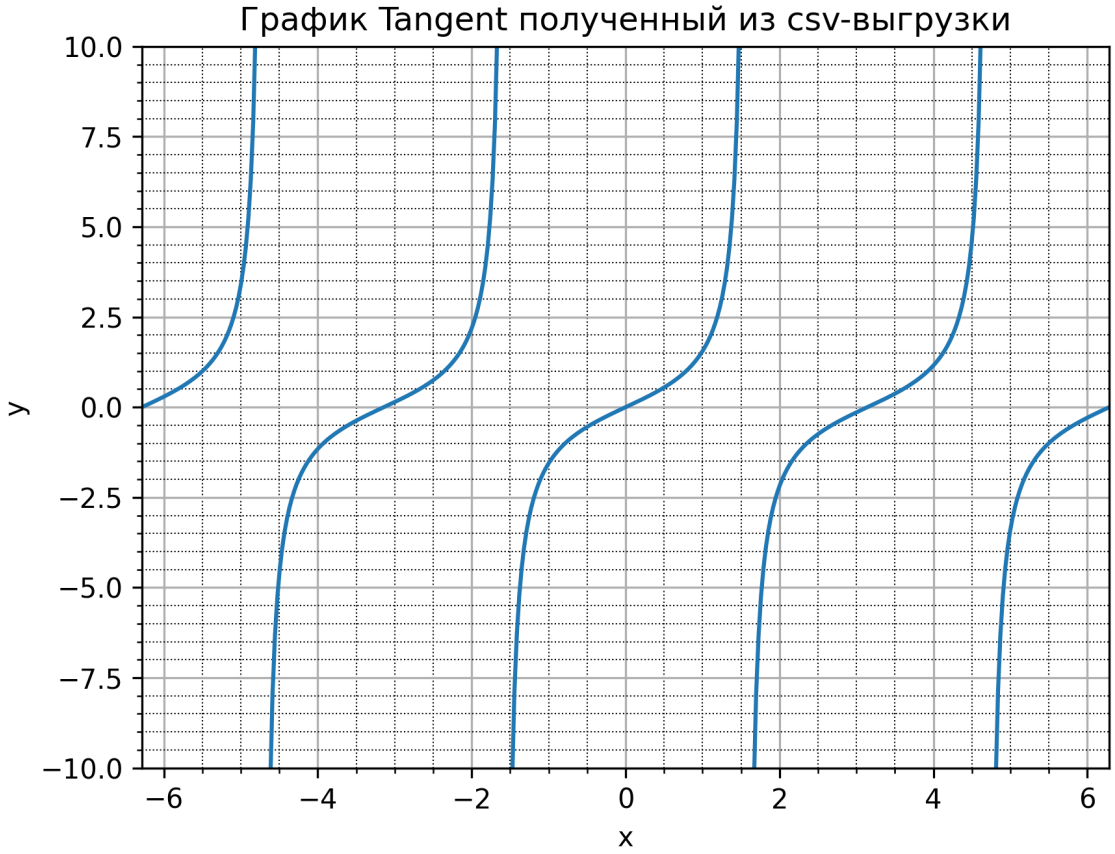


График Tangent полученный из csv-выгрузки
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
y
−6
−4
−2
0
2
4
6
x

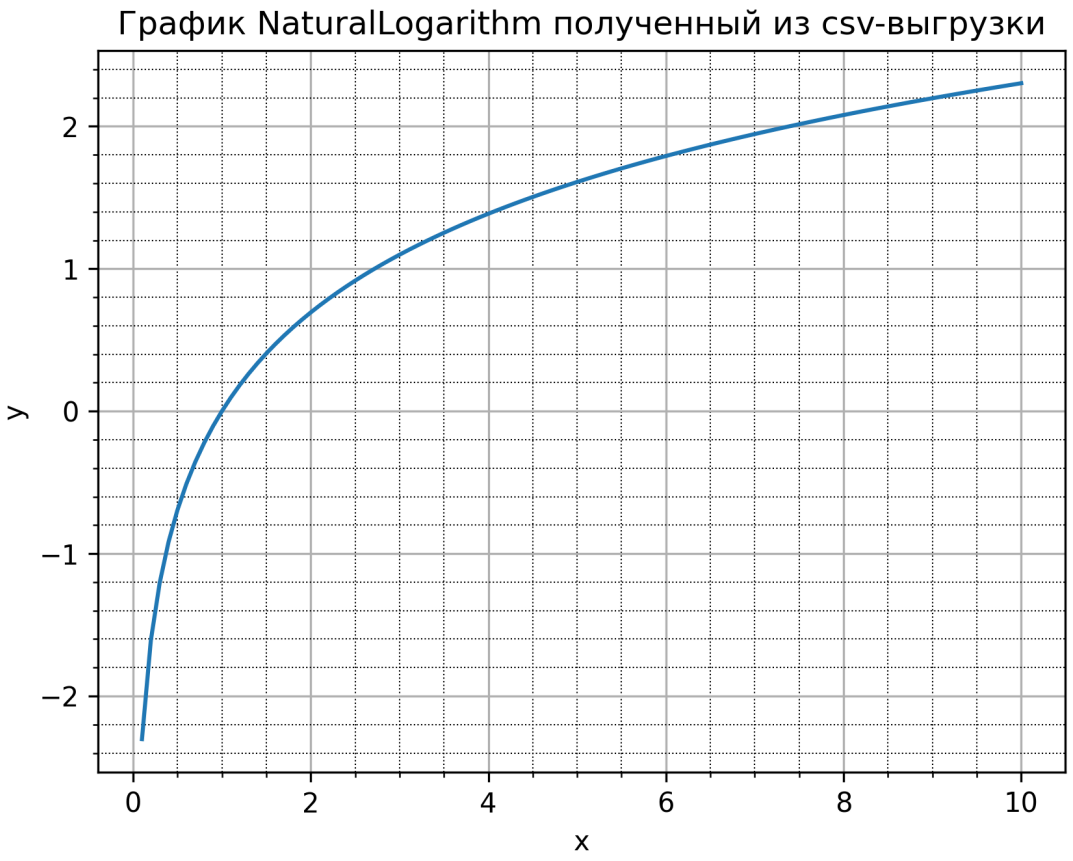


График NaturalLogarithm полученный из csv-выгрузки
y
x
2
1
0
−1
−2
0
2
4
6
8
10

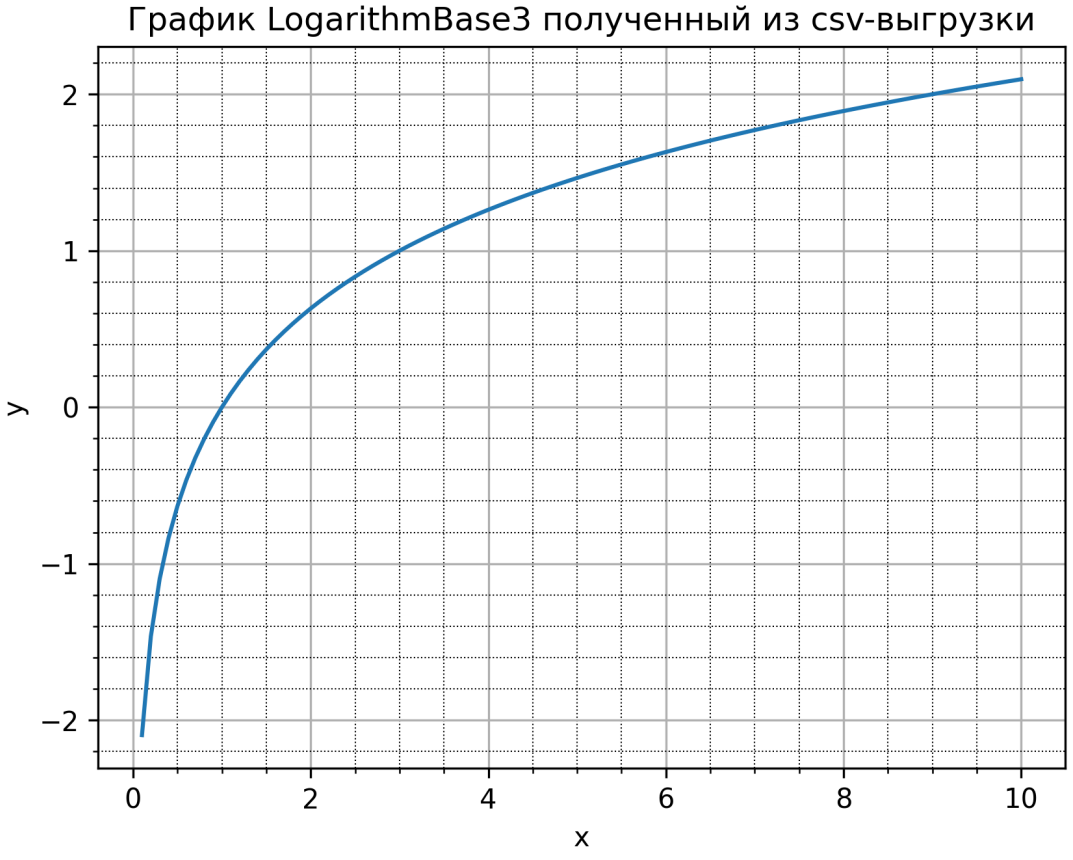
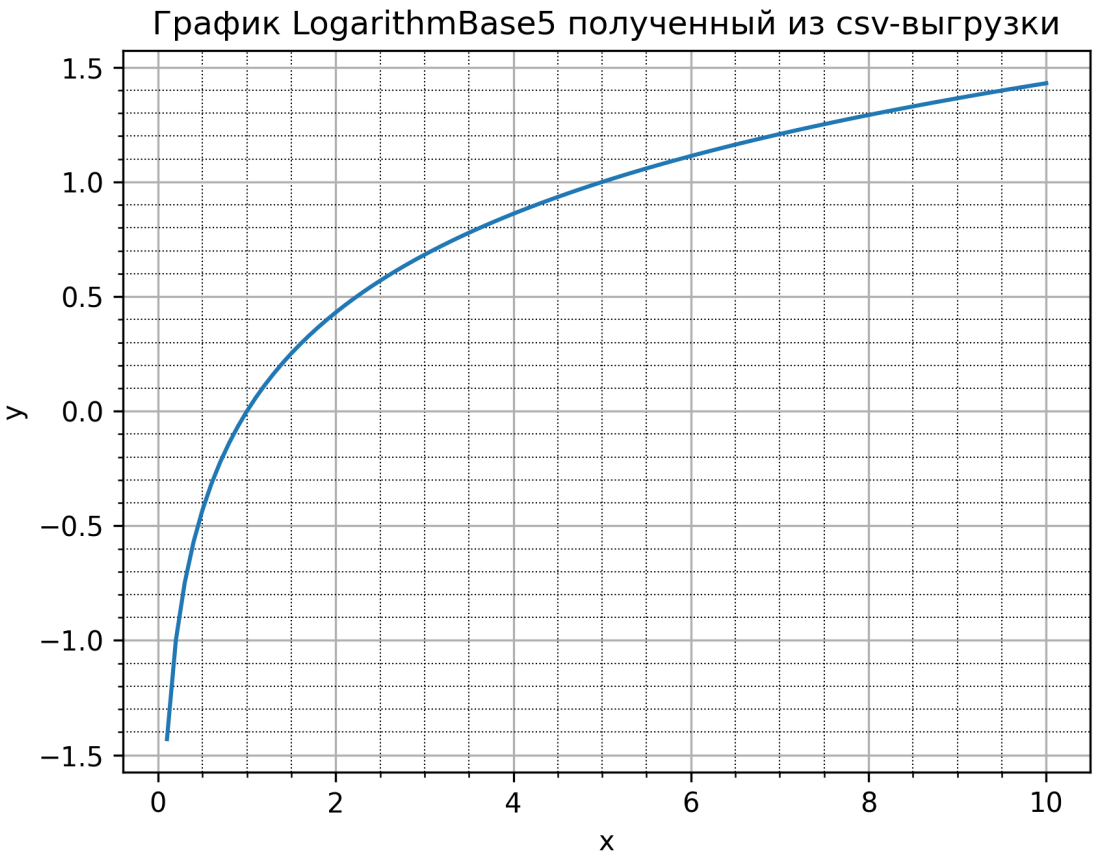


График LogarithmBase3 полученный из csv-выгрузки
y
x
2
1
0
−1
−2
0
2
4
6
8
10



Исходный код

Можно просмотреть в моём GitHub: https://github.com/AaLexUser/Software-testing

Выводы по работе.

В ходе выполнения данной лабораторной работы я провел интеграционное тестирование разработанной программы и изучил работу классов-заглушек на примере Mockito.