Университет ИТМО

Тестирование программного обеспечения

Лабораторная работа №1

Вариант 681

Выполнила: Калугина Марина

Группа: P3402

г. Санкт-Петербург

2020 г.

Задание

1. Для указанной функции провести модульное тестирование разложения функции в степенной ряд. Выбрать достаточное тестовое покрытие.
2. Провести модульное тестирование указанного алгоритма. Для этого выбрать характерные точки внутри алгоритма, и для предложенных самостоятельно наборов исходных данных записать последовательность попадания в характерные точки. Сравнить последовательность попадания с эталонной.
3. Сформировать доменную модель для заданного текста. Разработать тестовое покрытие для данной доменной модели

1. Функция arcsin(x)
2. Программный модуль для работы с АВЛ-деревом (http://www.cs.usfca.edu/~galles/visualization/AVLtree.html)
3. Описание предметной области:

> Англия больше не существует. Каким-то образом он смог это осознать. Он попытался еще раз. Америки больше нет, подумал он. Это не охватывалось. Нужно снова попробовать что-нибудь помельче. Нью-Йорка нет. Не получается. Он все равно никогда всерьез не верил, что он существует. Доллар упал навсегда. Что-то ощущается. Все фильмы с Богартом пропали, сказал он себе, и ему стало тоскливо. Макдональдс, вспомнил он. Больше никогда не будет маковских гамбургеров.

Описание используемого Junit

Используется Junit5 из `org.junit.jupiter.api.*`

Junit - библиотека для модульного тестирования в Java.

В JUnit Jupiter имеются следующие артефакты:

- junit-jupiter-api — JUnit Jupiter API для написания тестов и расширений;
- junit-jupiter-engine — реализация тестовой среды JUnit Jupiter, которая необходима только во время выполнения;
- junit-jupiter-params обеспечивает поддержку для параметризованных тестов в JUnit Jupiter;
- junit-jupiter-migrationsupport обеспечивает поддержку миграции с JUnit 4 на JUnit Jupiter и требуется только для запуска выбранных правил JUnit 4.

Используемые аннотации в лабораторной работе:

@Test - аннотация для объявления тестов

@BeforeEach - аннотация для выполнения метода перед каждым тестом

@AfterEach - аннотация для выполнения метода перед каждым тестом

Прочие аннотации:

@BeforeAll/@AfterAll - аннотация для выполнения метода перед каждым классом

@ParameterizedTest - аннотация для обозначения того, что что метод является параметризованным тестом

@Tag - аннотация для пометками тэгами тестов для возможности фильтрации

@Disabled - аннотация для отключения тестового сценария

@RepeatedTest - аннотация лдя обозначения того, что метод является естовым шаблоном для повторного теста. Такие методы наследуются, если они не переопределены.

@TestFactory - аннотация означает, что метод является тестовой фабрикой для динамических тестов.

@Nested - аннотация означает, что аннотированный класс является нестатическим вложенным тестовым классом.

@Timeout - используется для определения тайм-аута для метода или всех тестируемых методов

@TestTemplate - означает, что метод является шаблоном для тестовых случаев

@TestMethodOrder - используется для настройки порядка выполнения тестового метода

@TestInstance - используется для настройки жизненного цикла тестового экземпляра

@DisplayName - настраиваемое отображаемое имя для тестового класса или тестового метода.

@DisplayNameGeneration - объявляет настраиваемый генератор отображаемых имен для тестового класса.

Исходный код

https://github.com/KaluginaMarina/FourthYearOfItmo/tree/master/testing/lab1

Вывод

В ходе выполнения лабораторной работы были получены навыки модульного тестирования программного обеспечения при помощи библиотеки JUnit