САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине «Тестирование Программного Обеспечения»

Вариант 663

Преподаватель

Исаев Илья Владимирович

Выполнил

студент группы P3410

Возжаев. А. В

Санкт-Петербург – 2020

Задание:

1. Для указанной функции провести модульное тестирование разложения функции в степенной ряд. Выбрать достаточное тестовое покрытие.

Функция arcsin(x)

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.

Программный модуль для работы с АВЛ-деревом

3. Сформировать доменную модель для заданного текста. Разработать тестовое покрытие для данной доменной модели

Легко, как балерина, Зафод вскочил на ноги и начал осматриваться. До самого горизонта во все стороны простиралась сплошная золотая поверхность. Она блестела, как... впрочем, этому невозможно подобрать сравнение, потому что ничто во Вселенной не блестит так, как планета из чистого золота.

Описание Junit 4:

JUnit – Это регрессивный фреймворк, предназначенный для тестирования, который используется разработчиками для реализации юнит-тестирования при разработке на языке Java.

JUnit 4 объединяется в один файл jar.

Фреймворка JUnit состоит из следующих ключевых групп:

- Fixtures (каркас) - фиксированное состояние множества объектов, которые служат базисом для выполнения тестов. Гарантирует повторение результатов при проведении теста.
- Test Suites (группа тестов) - Это группа, состоящая из нескольких тестов, которые запускаются вместе. Для запуска групповых тестов используются аннотации @RunWith и @Suite.
- Test runners (сущности, которые выполняют тесты) - Используются для выполнения тестовых случаев.
- Классы JUnit - используются для написания тестов и их выполнения:
    - Assert - содержит множество методов утверждений.
    - TestCase - содержит тестовые случаи, который определяют каркас для выполнения нескольких тестов.

- TestResult - методы для хранения данных, полученных в результате выполнения тестовых случаев.

Аннотации:

@Test (использовалось**)**
Собственно сам тест

@Ignore
Игнорировать метод при тестировании

@Before (использовалось**)**
Выполняется перед запуском метода с аннотацией **@Test**. Используется для инициализации данных для последующего теста.
@After
Выполняется после завершения работы каждого теста. Подчищаем за собой.

@BeforeClass
Выполняется один раз перед запуском всех тестов. Используется для инициализации общих для всего теста данных
@AfterClass
Выполняется один раз после прохождения всех тестов.

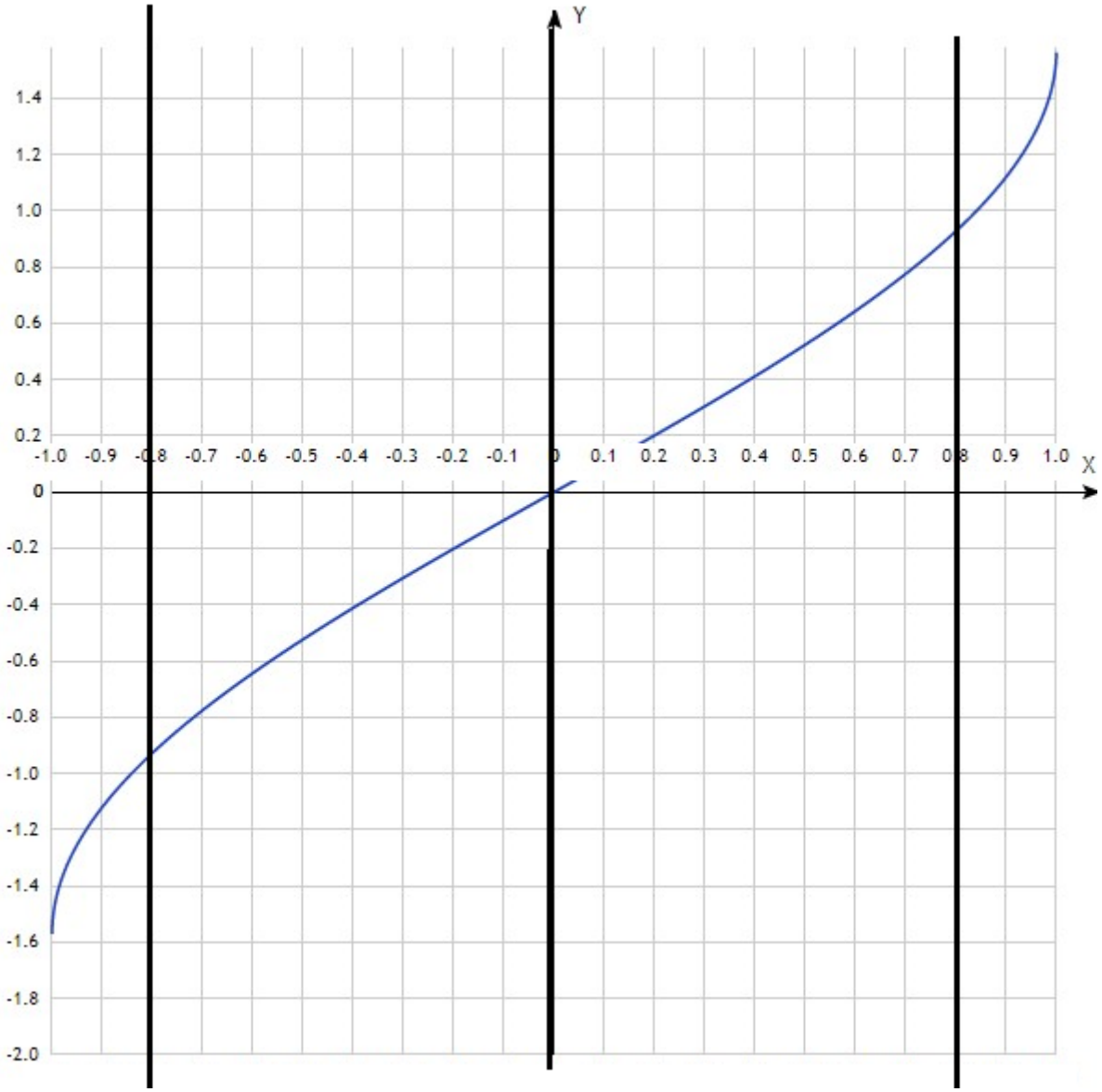
@Ignore - тест, помеченный данной аннотацией, будет пропускаться

@RunWith - если класс содержит данную аннотацию, JUnit будет запускать тесты в классе, который указан в параметрах этой аннотации

@Category - разновидность раннера, который запускает только те тестовые классы и методы, которые аннотированы @IncludeCategory

@Parameter - статический метод, содержащий данную аннотацию создает и возвращает коллекцию из массивов элементов, которые являются параметрами для тестового метода.

1) Функция arcsin(x)



Разложение функции в степенной ряд производится по след. формуле:

$$\arcsin x = x + \frac{x^3}{2\cdot 3} + \frac{1\cdot 3 x^5}{2\cdot 4\cdot 5} + \ldots + \frac{1\cdot 3\cdot 5\ldots(2n-1)x^{2n+1}}{2\cdot 4\cdot 6\ldots(2n)(2n+1)} + \ldots, \quad |x| < 1$$

arcsin(x)+arccos(x)=pi/2

arccos(x)=arcsin(sqrt(1-xx)) - формулы тригонометрии

arcsin(x)=pi/2-arcsin(sqrt(1-xx))

Выбор тестового покрытия, используя классы эквивалентности:

Получим отрезки (-infinity;-1), [-1; -0,8], (-0.8; 0], (0; 0,8], (0,8, 1], (1, +infinity)

Внутри промежутка для теста 2 значения

Соответственно получим 12 тестовых случаев внутри промежутков и 6 на концах отрезков и того тестовое покрытие 18 тестовых случаев.

2) Программный модуль для работы с АВЛ-деревом

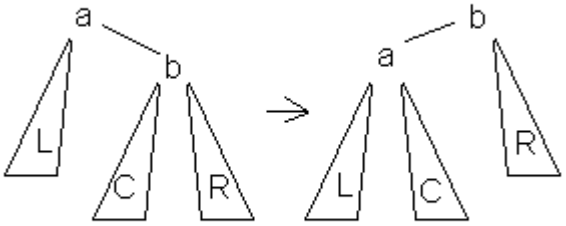
АВЛ-дерево — сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

Коэффициент балансировки узла N равен высоте (справа (N)) - высоте (слева (N)). В дереве AVL коэффициент баланса узла может быть только одним из значений 1, 0 или -1.

Для соблюдения этого условия используется балансировка

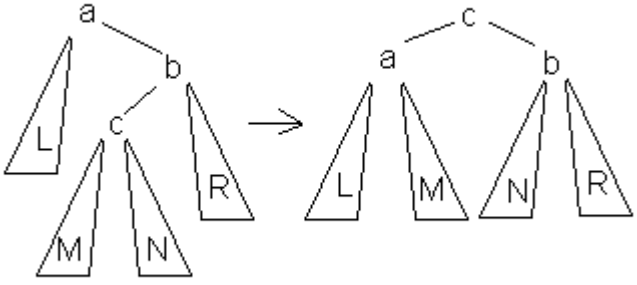
Балансировкой вершины называется операция, которая в случае разницы высот левого и правого поддеревьев $|h(L)-h(R)|=2$, изменяет связи предок-потомок в поддереве данной вершины так, чтобы восстановилось свойство дерева $|h(L)-h(R)|\leqslant 1$, иначе ничего не меняет. Для балансировки будем хранить для каждой вершины разницу между высотой её левого и правого поддерева $diff[i]=h(L)-h(R)$

Малое левое вращение



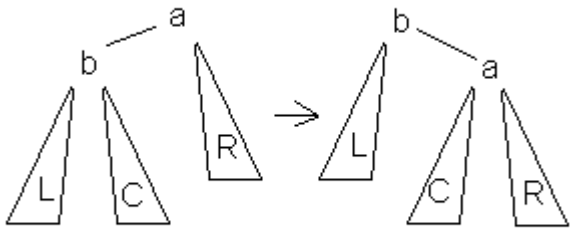
Данное вращение используется тогда, когда (высота b-поддерева — высота L) = 2 и высота c-поддерева <= высота R.

Большое левое вращение



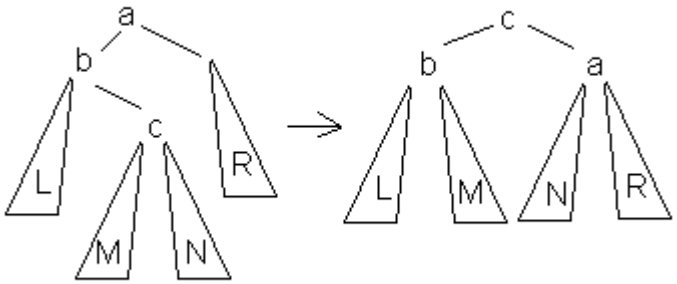
Данное вращение используется тогда, когда (высота b-поддерева — высота L) = 2 и высота c-поддерева > высота R.

Малое правое вращение



Данное вращение используется тогда, когда (высота b-поддерева — высота R) = 2 и высота C <= высота L.

Большое правое вращение



Данное вращение используется тогда, когда (высота b-поддерева — высота R) = 2 и высота c-поддерева > высота L.

Программный модуль для работы с AVL деревом поддерживает операции вставки и удаления.

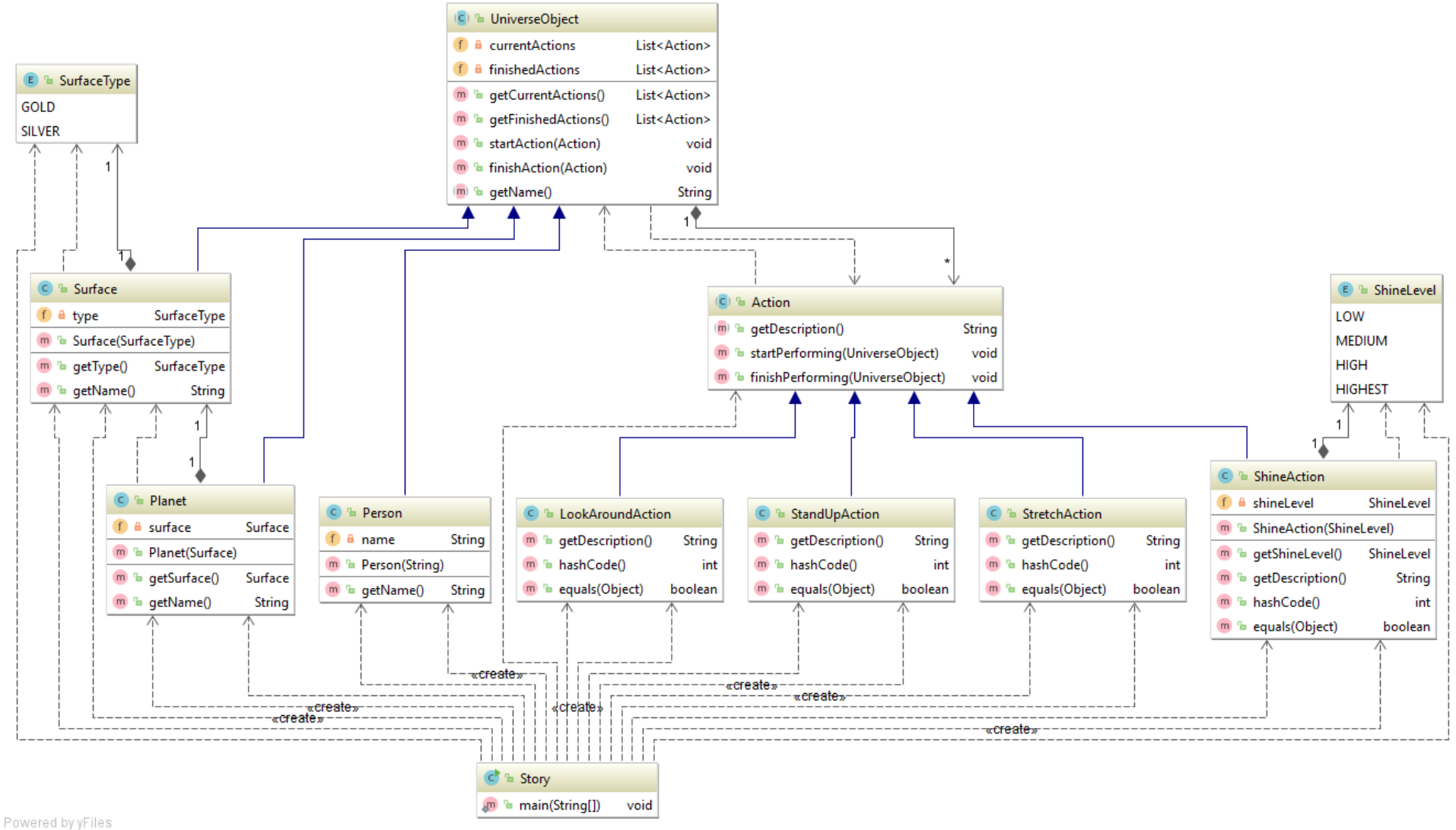
Тестирование:

Были выбраны следующие характерны точки внутри алгоритма:

- Добавление в дерево единственной вершины
- Удаление из дерева единственной вершины
- Создание дерева с множеством узлов
- Удаление всех узлов

- Удаление наименьшего узла
- Удаление наибольшего узла
- Удаление несуществующего узла
- Балансировка – малый левый поворот (добавление в правый правый узел)
- Балансировка – малый правый поворот (добавление в левый левый узел)
- Балансировка – большой левый поворот (добавление в правый левый узел)
- Балансировка – большой правый поворот (добавление в правый левый правый узел)
- Проверка высоты пустого дерева
- Проверка высоты дерева с 1 элементом

3) Доменная модель по тексту



Исходный код:
https://github.com/artem8449/TPO_lab1

Вывод:
В ходе выполнения данной лабораторной работы были получены навыки модульного тестирования при помощи фреймворка Junit 4. Данный фреймворк предоставляет удобные средства тестирования, и в отличии от Junit 3, он использует аннотации. Junit 5 в отличии от Junit 4 имеет иную иерархию классов и модульность, это делает его более гибким. Так же были добавлены аннотации, а часть претерпели изменеия.