Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по дисциплине
«Тестирование программного обеспечения»

Вариант на 235726

Студент:
Группа № P33101 *Павлова А.И.*

Предподаватель: Машина.Е.А.

г.Санкт-Петербург 2024

Содержание

1 Задание лабораторной работы

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

1.1 Данные по варианту

1. Функция $sin(x)$
2. Программный модуль для работы с B+ деревьями (максимальное количество элементов в ключе - 6) http://www.cs.usfca.edu/~galles/visualization/BPlusTree.html
3. Описание предметной области:

   В первый момент показалось, что ничего не произошло, затем что-то засветилось на краю огромного экрана. По нему ползла красная звезда величиной с тарелку, а следом за ней еще одна: бинарная звездная система. Затем в углу картинки возник большой полумесяц – красный свет, переходящий в черноту – ночная сторона планеты.

2 Процесс выполнения

2.1 Модульное тестирвоание функции

Создадим класс task1, в котормо пропишем методы для вычиления функции по Тейлору

```java
package main;


public class task1 {
    // https://function-x.ru/chapter9-4/rows4_clip_image063.gif
    public static double correctX(double x){
        double PI = Math.PI;

        if (x >= 0) {
            while (x > PI) {
                x -= PI;
            }
        } else if (x < 0){
            while (x < -PI) {
                x += PI;
            }
        }
        return x;
    }

    public static int calculateFactorial(int n){
        int result = 1;
        for (int i = 1; i <=n; i++){
            result = result*i;
        }
        return result;
    }

    public static double commonTaylor(double x, int n) {
        if (n % 2 == 0) {
            return  -(Math.pow(x, 2*n-1) / calculateFactorial(2*n-1));  //

        } else {
            return (Math.pow(x, 2*n-1)) / calculateFactorial(2*n-1); //

        }
    }

    public static double sinTaylor(double x, int n){
        double result = 0; //
        double newX = correctX(x);

```

```
        for (int i = 1; i < n; i += 1){
            result += commonTaylor(newX, i);
        }
        return result;
    }

}
```

Листинг 1: Функция $sin(x)$

Тперь создаим класс, который будет тестировать нашу функцию. Так как Тейлор выдает приблизительное значение нашей функции, будем сравнивать значения с некоторой погрешностью. Выставим значение погрешности $1e-5$ это 0.00001.

```
package test;
import main.task1;
import org.junit.Test;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;
import org.junit.jupiter.params.provider.MethodSource;

import java.util.stream.Stream;

import static org.junit.Assert.assertEquals;
import static org.junit.jupiter.api.Assertions.assertAll;


public class task1test{
    private static Stream<org.junit.jupiter.params.provider.Arguments> sinValues() {
        return Stream.of(
                org.junit.jupiter.params.provider.Arguments.of(-Math.PI / 4, -Math.sqrt(2) / 2),
                org.junit.jupiter.params.provider.Arguments.of(0, 0.0),
                org.junit.jupiter.params.provider.Arguments.of(Math.PI / 4, Math.sqrt(2) / 2),
                org.junit.jupiter.params.provider.Arguments.of(Math.PI / 6, 0.5),
                org.junit.jupiter.params.provider.Arguments.of(Math.PI / 3, Math.sqrt(3) / 2)
        );
    }

    @ParameterizedTest(name = "sin(x)=1")
    @DisplayName("Check n * PI/2  dots")
    @ValueSource(doubles = {Math.PI / 2, 2 *Math.PI + Math.PI / 2, -1 * Math.PI - Math.PI / 2})
    void test1(double param){
        assertAll(
                () -> assertEquals(1, task1.sinTaylor( param, 10), 1e-4)
        );
    }
    @ParameterizedTest(name = "sin(x)= -1")
    @DisplayName("Check n * PI/2  dots")
    @ValueSource(doubles = {-Math.PI / 2, -(2 *Math.PI + Math.PI / 2), Math.PI + Math.PI / 2, -(6*
    Math.PI + Math.PI / 2)})
    void test2(double param){
        assertAll(
                () -> assertEquals(-1, task1.sinTaylor( param, 10), 1e-5)
        );
    }

    @ParameterizedTest(name = "sin(x) other")
    @DisplayName("Check dots in range(-1; 1)")
    @MethodSource("sinValues")
    void test3(double param, double expectedValue){
        assertAll(
                () -> assertEquals(expectedValue, task1.sinTaylor( param, 10), 1e-5)
        );
    }
}
```

Листинг 2: Тесты для функции

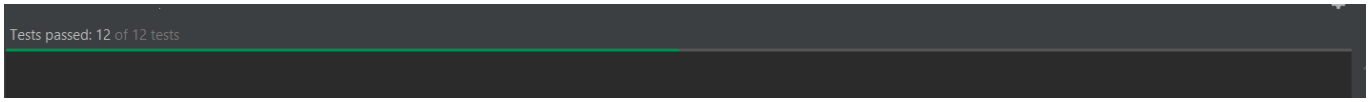

Рис. 1: Консоль с успешным прохождением тестов

2.2 Модульное тестирование алгоритма

B+ деревья - это структура данных, представляющая собой сбалансированное n-арное дерево поиска, которое является модификацией B-дерева. Основное отличие B-дерева заключается в том, что все ключи и сопутствующие данные хранятся только в листьях, а во внутренних узлах находятся только копии ключей.

- Все значения находятся в листьях дерева
- Имеет огрничение на количество потомков и количества значений в узле
- Внутренние узлы служат для навигации по дереву, а не для хранения данных

Реализация моего алгоритма отличается от того, что был предложен для сравнения , но он не науршает правил составления дерева так что тоже верен.
Привожу пример для дерева с ограничением на 3 потомков:

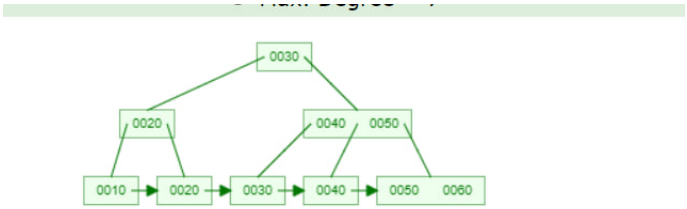

Рис. 2: Дерево составленное предложенным ресурсом

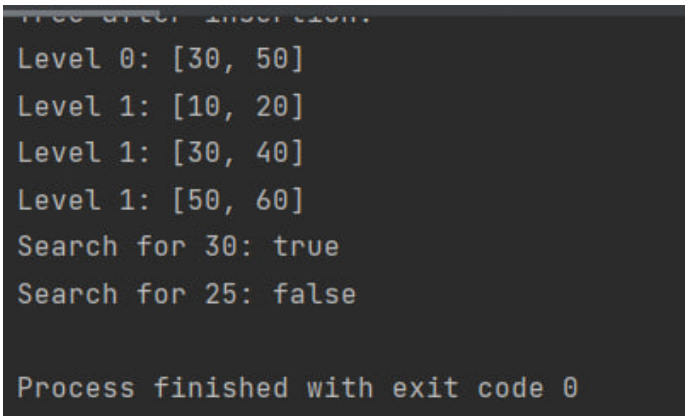

Рис. 3: Дерево составленное программой

2.3 Доменная модель и тествоео покрытие

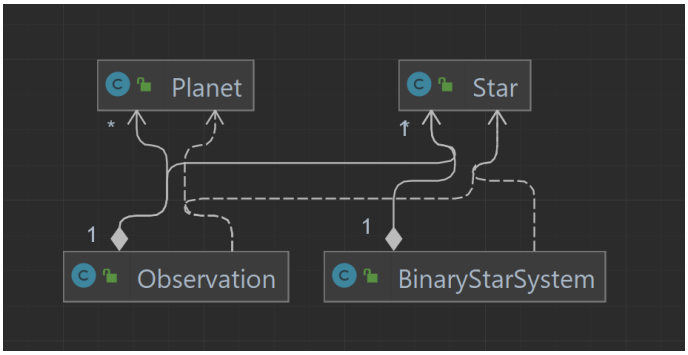

Рис. 4: Диграмма доменной модели

Мною были выделены классы звезда, бинарная звездная система, наблюдениеи планета. Тестирование проходили изменение полей и вывод полей

3 Вывод

В данной лабораторной работе мно были изучен фреймворк JUnit и аннотации для написнаия тестов.