

ЛАБОРАТОРНАЯ РАБОТА № 1
по дисциплине
«Тестирование программных систем»

Вариант №-31214

Выполнил:
Студент группы Р33312
Гончаров Андрей
Викторович
Мартынов Всеволод
Владимирович
Преподаватель: Наумова
Надежда Александровна

ЗАДАНИЕ

Для выполнения лабораторной работы №1 необходимо:

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

Введите вариант:

1. Функция $\text{tg}(x)$
2. Программный модуль для работы с хеш-таблицей с закрытой адресацией (Hash String, <http://www.cs.usfca.edu/~galles/visualization/OpenHash.html>)
3. Описание предметной области:

Артур, нервничая, вошел следом и был ошеломлен, увидев развалившегося в кресле человека, положившего ноги на пульт управления и ковыряющего левой рукой в зубах правой головы. Правая голова, казалось, была всецело занята этим, но зато левая улыбалась широко и непринужденно. Количество вещей, видя которые, Артур не верил своим глазам, все росло. Его челюсть отвисла.

ПУНКТ 1

Разработанный модуль для вычисления тангенса:

```
package ifmo.block1;

public class Tan {
    public static double calcTan(double x, int n) {
        if (Double.isInfinite(x) || Double.isNaN(x)) {
            throw new IllegalArgumentException("Argument can't be equal to
infinite or null!");
        } else if (Math.abs(x) % (Math.PI) == Math.PI / 2) {
            throw new IllegalArgumentException("tg(" + x + ") is equal to
infinity!");
        }
        double sin = calcSin(x, n);
        double cos = calcCos(x, n);
        return sin / cos;
    }

    private static double calcSin(double x, int n) {
        x = x % (Math.PI * 2);
        double result = x;
        double x2 = x * x;
        double pow = x;
        double fact = 1;
        int sign = -1;
        for (int i = 1; i < n; i++) {
            fact *= 2 * i * (2 * i + 1);
            pow *= x2;
            result += sign * pow / fact;    // (-1)^(n) * x^(2n+1) / (2n+1)!
            sign = -sign;
        }




        return result;
    }

    private static double calcCos(double x, int n) {
        x = x % (Math.PI * 2);
        double result = 1;
        double x2 = x * x;
        double pow = 1;
        double fact = 1;
        int sign = -1;
        for (int i = 1; i < n; i++) {
            fact *= 2 * i * (2 * i - 1);
            pow *= x2;
            result += sign * pow / fact;    // (-1)^(n) * x^(2n) / (2n)!
            sign = -sign;
        }

        return result;
    }
}
```

Test coverage:

ifmo.block1

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 Tan.java		97%		100%	1	9	1	32	1	4	0	1
Total	3 of 147	97%	0 of 10	100%	1	9	1	32	1	4	0	1

```
1. package ifmo.block1;
2.
3. public class Tan {
4.     public static double calcTan(double x, int n) {
5.         if (Double.isInfinite(x) || Double.isNaN(x)) {
6.             throw new IllegalArgumentException("Argument can't be equal to infinite or null!");
7.         } else if (Math.abs(x) % (Math.PI) == Math.PI / 2) {
8.             throw new IllegalArgumentException("tg(" + x + ") is equal to infinity!");
9.         }
10.        double sin = calcSin(x, n);
11.        double cos = calcCos(x, n);
12.        return sin / cos;
13.    }
14.
15.
16.    private static double calcSin(double x, int n) {
17.        x = x % (Math.PI * 2);
18.        double result = x;
19.        double x2 = x * x;
20.        double pow = x;
21.        double fact = 1;
22.        int sign = -1;
23.        for (int i = 1; i < n; i++) {
24.            fact *= 2 * i * (2 * i + 1);
25.            pow *= x2;
26.            result += sign * pow / fact; // (-1)^(n) * x^(2n+1) / (2n+1)!
27.            sign = -sign;
28.        }
29.
30.        return result;
31.    }
32.
33.    private static double calcCos(double x, int n) {
34.        x = x % (Math.PI * 2);
35.        double result = 1;
36.        double x2 = x * x;
37.        double pow = 1;
38.        double fact = 1;
39.        int sign = -1;
40.        for (int i = 1; i < n; i++) {
41.            fact *= 2 * i * (2 * i - 1);
42.            pow *= x2;
43.            result += sign * pow / fact; // (-1)^(n) * x^(2n) / (2n)!
44.            sign = -sign;
45.        }
46.        return result;
47.    }
48. }
```

ПУНКТ 2

Модуль для hash таблицы:

```
package ifmo.block2;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class HashMap {
    private static final int DEFAULT_CAPACITY = 12;
    private ArrayList<LinkedList<String>> hashArray;

    public HashMap() {
        this.hashArray = new ArrayList<>(DEFAULT_CAPACITY);
        for (int i = 0; i < DEFAULT_CAPACITY; i++) {
            hashArray.add(new LinkedList<>());
        }
    }

    private int hash(String key) {
        int hash = 0;
        for (int i = 0; i < key.length(); i++) {
            hash ^= key.charAt(i); // bitwise XOR operation for each
character
        }
        return Math.abs(hash % DEFAULT_CAPACITY);
    }

    public void put(String key) {
        int index = hash(key);
        LinkedList<String> list = hashArray.get(index);
        list.add(key);
    }


    public void remove(String key) {
        int index = hash(key);
        LinkedList<String> list = hashArray.get(index);
        list.remove(key);
    }

    public boolean contains(String key) {
        int index = hash(key);
        LinkedList<String> list = hashArray.get(index);
        return list.contains(key);
    }

    public List<String> get(String key) {
        int index = hash(key);
        LinkedList<String> list = hashArray.get(index);
        return list;
    }
}
```

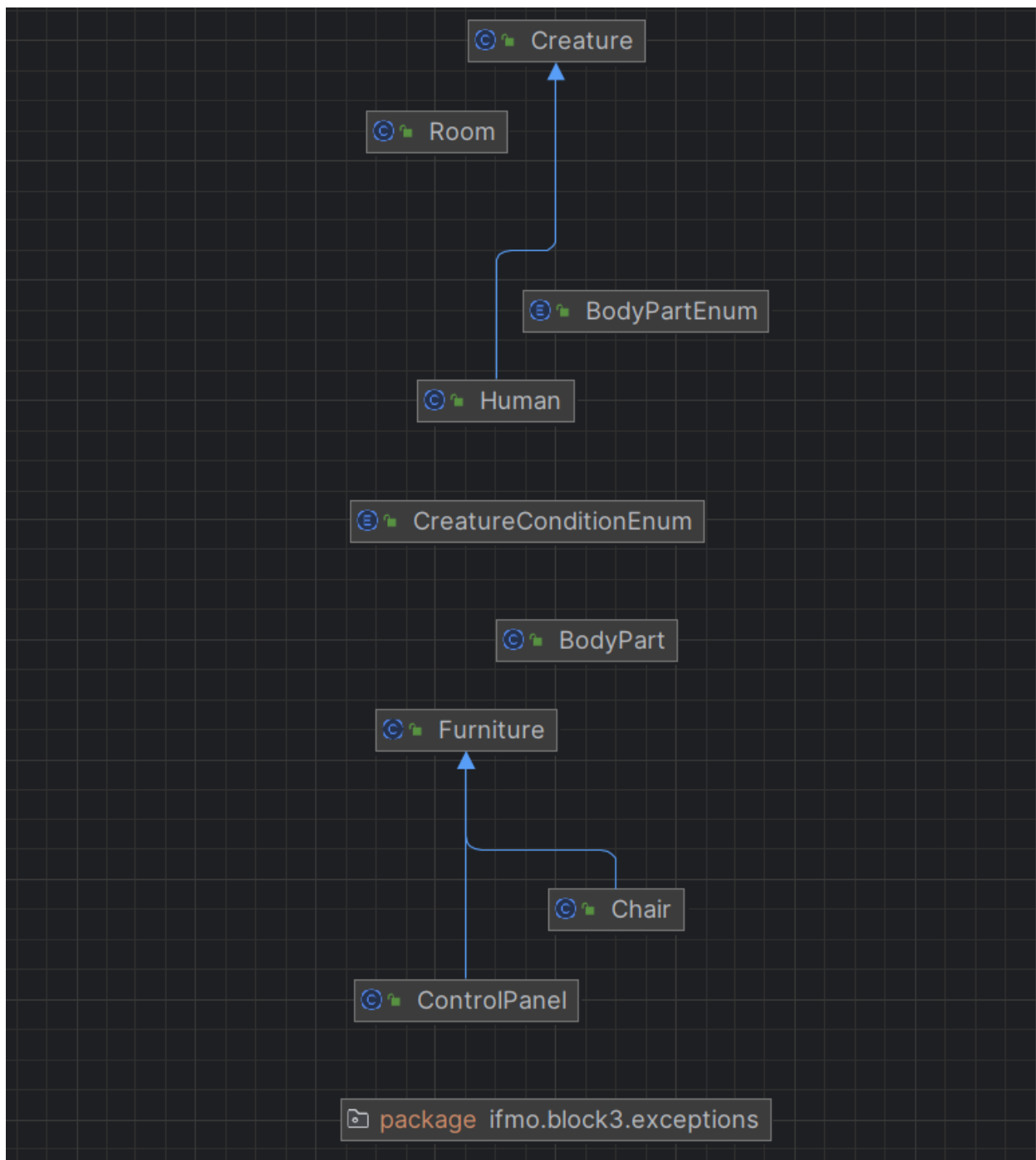
Test coverage:

ifmo.block2

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 HashMap	<div><div></div></div>	100%	<div><div></div></div>	100%	0	8	0	23	0	6	0	1
Total	0 of 100	100%	0 of 4	100%	0	8	0	23	0	6	0	1














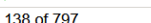
ПУНКТ 3

UML диаграмма разработанной объектной модели:

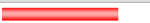
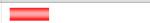
















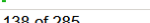


Test coverage:

ifmo.block3

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
BodyPart		51%		39%	20	39	0	25	2	15	0	1
Human		100%		100%	0	11	0	33	0	7	0	1
Room		100%		100%	0	23	0	24	0	11	0	1
Creature		100%		100%	0	14	0	27	0	8	0	1
Furniture		100%		100%	0	11	0	12	0	5	0	1
CreatureConditionEnum		100%		n/a	0	1	0	6	0	1	0	1
BodyPartEnum		100%		n/a	0	1	0	5	0	1	0	1
Chair		100%		n/a	0	5	0	10	0	5	0	1
ControlPanel		100%		n/a	0	1	0	2	0	1	0	1
Total	138 of 797	82%	29 of 104	72%	20	106	0	144	2	54	0	9

BodyPart

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
hashCode()		0%		0%	5	5	1	1	1	1
equals(Object)		47%		30%	14	16	0	1	0	1
toString()		0%		n/a	1	1	1	1	1	1
endInteraction(BodyPart)		100%		100%	0	4	0	9	0	1
interact(BodyPart, String)		100%		100%	0	3	0	8	0	1
BodyPart(String, int, int, BodyPartEnum)		100%		n/a	0	1	0	2	0	1
setInAction(boolean)		100%		n/a	0	1	0	1	0	1
setInteractionBodyPart(BodyPart)		100%		n/a	0	1	0	1	0	1
getName()		100%		n/a	0	1	0	1	0	1
getLength()		100%		n/a	0	1	0	1	0	1
getWeight()		100%		n/a	0	1	0	1	0	1
getBodyPartEnum()		100%		n/a	0	1	0	1	0	1
isInAction()		100%		n/a	0	1	0	1	0	1
getInteractionBodyPart()		100%		n/a	0	1	0	1	0	1
canEqual(Object)		100%		n/a	0	1	0	1	0	1
Total	138 of 285	51%	29 of 48	39%	20	39	0	25	2	15

100% покрытие отсутствует только на методах, сгенерированных Lombok с помощью аннотации @Data

ВЫВОД

За лабораторную работу мы познакомились с Junit 5, Jасосо. Научились писать тесты и добились test coverage 100%.