

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!");
        }
        if ((int) (Math.abs(x / (Math.PI / 2))) % 2 == 1) {
            x = x % (Math.PI / 2);
            if (x >= 0) {
                x -= (Math.PI / 2);
            } else {
                x += (Math.PI / 2);
            }
        } else {
            x = x % (Math.PI / 2);
        }
        double sum = 0;
        for (int i = 1; i <= n; i++) {
            sum += ((Math.pow(-1, i + 1) * Math.pow(2, 2 * i) * (Math.pow(2,
2 * i) - 1) * BernoulliNumber.computeBernoulliNumber(2 * i))
                / factorial(2 * i).doubleValue()) * Math.pow(x, 2 * i -
1);
        }
        return sum;
    }

    private static BigInteger factorial(int n) {
        BigInteger result = BigInteger.ONE;
        for (int i = 2; i <= n; i++) {
            result = result.multiply(BigInteger.valueOf(i));
        }
        return result;
    }
}
```

```

public class BernoulliNumber {
    public static double computeBernoulliNumber(int N) {

        if (N > 1 && N % 2 == 1) return 0;
        N++;
        BigInteger[][] bin = new BigInteger[N + 1][N + 1];

        for (int i = 1; i <= N; i++) {
            bin[0][i] = BigInteger.ZERO;
        }
        for (int i = 0; i <= N; i++) {
            bin[i][0] = BigInteger.ONE;
        }
        for (int i = 1; i <= N; i++) {
            for (int j = 1; j <= N; j++) {
                bin[i][j] = bin[i - 1][j - 1].add(bin[i - 1][j]);
            }
        }

        double[] bernoulliNumbers = new double[N + 1];
        bernoulliNumbers[0] = 1;
        bernoulliNumbers[1] = -0.5;
        for (int i = 2; i < N; i++) {
            bernoulliNumbers[i] = 0;
            for (int j = 0; j < i; j++) {
                BigInteger coef = bin[i + 1][i + 1 - j];
                bernoulliNumbers[i] = bernoulliNumbers[i] -
(coef.doubleValue() * bernoulliNumbers[j]);
            }
            bernoulliNumbers[i] = bernoulliNumbers[i] / (i + 1);
        }
        return bernoulliNumbers[N - 1];
    }
}

```

Тесты:

```

class TanTest {

    @Test
    @DisplayName("Check illegal inputs")
    void checkWrongInputs() {
        assertAll(
            () -> assertThrows(IllegalArgumentException.class, () ->
Tan.calcTan(Double.POSITIVE_INFINITY, 50)),
            () -> assertThrows(IllegalArgumentException.class, () ->
Tan.calcTan(Double.NEGATIVE_INFINITY, 50)),
            () -> assertThrows(IllegalArgumentException.class, () ->
Tan.calcTan(Double.NaN, 50)),
            () -> assertThrows(IllegalArgumentException.class, () ->
Tan.calcTan(Math.PI / 2, 50)),
            () -> assertThrows(IllegalArgumentException.class, () ->
Tan.calcTan(5 * Math.PI / 2, 50)),
            () -> assertThrows(IllegalArgumentException.class, () ->
Tan.calcTan(-Math.PI / 2, 50))
        );
    }

    @ParameterizedTest(name = "tan({0})")
    @DisplayName("Check PI")
    @ValueSource(doubles = {

```

```

        -Math.PI + 0.3 * Math.PI,
        -Math.PI + 0,
        -Math.PI - 0.3 * Math.PI,

        -0.3 * Math.PI,
        0,
        0.3 * Math.PI,

        Math.PI - 0.3 * Math.PI,
        Math.PI + 0,
        Math.PI + 0.3 * Math.PI,

    })
    void checkPiDots(double param) {
        assertAll(
            () -> assertEquals(Math.tan(param), Tan.calcTan(param, 50),
10e-4)
        );
    }

    @ParameterizedTest(name = "tan({0}) = {1}")
    @DisplayName("Check dots with high accuracy")
    @CsvFileSource(resources = "/tan_test.csv", numLinesToSkip = 1, delimiter
= ';')
    void checkBetweenDotsFromCsvHighAccuracy(double x, double y) {
        assertAll(
            () -> assertEquals(y, Tan.calcTan(x, 93), 10e-7)
        );
    }
}

```

Test coverage:

ifmo.block1

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|---------------------------------|------------------------|------|------------------------|------|--------|------|--------|-------|--------|---------|--------|---------|
| BernoulliNumber | <div><div></div></div> | 97% | <div><div></div></div> | 100% | 1 | 8 | 1 | 20 | 1 | 2 | 0 | 1 |
| Tan | <div><div></div></div> | 97% | <div><div></div></div> | 100% | 1 | 10 | 1 | 20 | 1 | 3 | 0 | 1 |
| Total | 6 of 278 | 97% | 0 of 26 | 100% | 2 | 18 | 2 | 40 | 2 | 5 | 0 | 2 |

Tan

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods |
|--------------------------------------|------------------------|------|------------------------|------|--------|------|--------|-------|--------|---------|
| Tan() | <div><div></div></div> | 0% | <div><div></div></div> | n/a | 1 | 1 | 1 | 1 | 1 | 1 |
| calcTan(double, int) | <div><div></div></div> | 100% | <div><div></div></div> | 100% | 0 | 7 | 0 | 15 | 0 | 1 |
| factorial(int) | <div><div></div></div> | 100% | <div><div></div></div> | 100% | 0 | 2 | 0 | 4 | 0 | 1 |
| Total | 3 of 131 | 97% | 0 of 14 | 100% | 1 | 10 | 1 | 20 | 1 | 3 |

BernoulliNumber

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods |
|---|------------------------|------|------------------------|------|--------|------|--------|-------|--------|---------|
| BernoulliNumber() | <div><div></div></div> | 0% | <div><div></div></div> | n/a | 1 | 1 | 1 | 1 | 1 | 1 |
| computeBernoulliNumber(int) | <div><div></div></div> | 100% | <div><div></div></div> | 100% | 0 | 7 | 0 | 19 | 0 | 1 |
| Total | 3 of 147 | 97% | 0 of 12 | 100% | 1 | 8 | 1 | 20 | 1 | 2 |

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) {
        if (key == null) {
            throw new IllegalArgumentException();
        }
        int index = hash(key);
        LinkedList<String> list = hashArray.get(index);
        list.add(key);
    }

    public void remove(String key) throws NoSuchElementException {
        if (key == null) {
            throw new IllegalArgumentException();
        }
        int index = hash(key);
        LinkedList<String> list = hashArray.get(index);
        if (list.isEmpty()) {
            throw new NoSuchElementException("No such element");
        }
        list.remove(key);
    }

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

    public List<String> get(String key) throws NoSuchElementException {
        if (key == null) {
```

```

        throw new IllegalArgumentException();
    }
    int index = hash(key);
    LinkedList<String> list = hashArray.get(index);
    if (list.isEmpty()) {
        throw new NoSuchElementException("No such element");
    }
    return list;
}
}

```

Тесты:

```

public class HashMapTest {
    private HashMap map;

    @BeforeEach
    public void setUp() {
        map = new HashMap();
    }

    @Test
    public void testPutAndGet() throws NoSuchElementException {
        map.put("apple");
        map.put("banana");
        map.put("orange");

        assertTrue(map.contains("apple"));
        assertTrue(map.contains("banana"));
        assertTrue(map.contains("orange"));
        assertFalse(map.contains("grape"));

        List<String> list = map.get("apple");
        assertNotNull(list);
        assertTrue(list.contains("apple"));
    }

    @Test
    public void testRemove() throws NoSuchElementException {
        map.put("apple");
        map.put("banana");
        map.put("orange");

        assertTrue(map.contains("banana"));
        map.remove("banana");
        assertFalse(map.contains("banana"));
    }

    @Test
    public void testEmptyMap() {
        assertFalse(map.contains("apple"));
        assertFalse(map.contains("banana"));
        assertFalse(map.contains("orange"));
    }

    @Test
    public void testNegativeCases() {
        assertThrows(IllegalArgumentException.class, () -> map.put(null));
    }
}

```


```

        assertThrows(NoSuchElementException.class, ()-> map.get("mango"));
        assertThrows(NoSuchElementException.class, ()->
map.remove("banana"));
        assertThrows(IllegalArgumentException.class, ()-> map.get(null));
        assertThrows(IllegalArgumentException.class, ()-> map.remove(null));
        assertThrows(IllegalArgumentException.class, ()->
map.contains(null));
    }
}

```

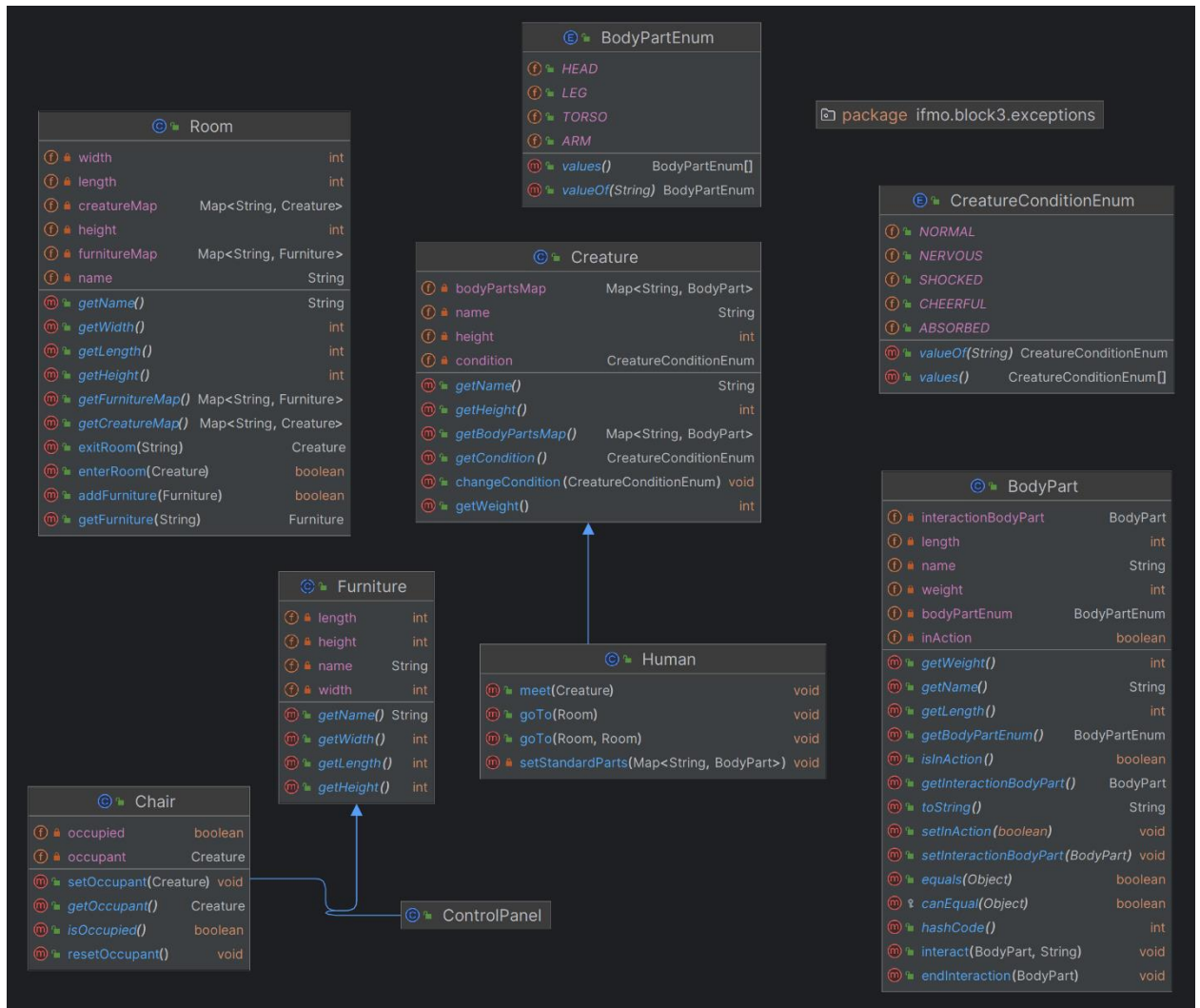
Test coverage:

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%.