"Национальная научно-образовательная корпорация ИТМО"

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

по дисциплине

«Тестирование программных систем»

Вариант №-31214

Выполнил:

Студент группы P33312

Гончаров Андрей Викторович

Мартынов Всеволод Владимирович

Преподаватель: Наумова Надежда Александровна



Санкт-Петербург, 2024

**ЗАДАНИЕ**

Для выполнения лабораторной работы №1 необходимо:

A close up of text

Description automatically generated**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:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**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:

**A screenshot of a computer

Description automatically generated**

**ПУНКТ 3**

UML диаграмма разработанной объектной модели:

A screenshot of a computer

Description automatically generated

Test coverage:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

**ВЫВОД**

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