"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!");  
 }  
 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:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

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){  
 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:

**A screenshot of a computer

Description automatically generated**

**ПУНКТ 3**

UML диаграмма разработанной объектной модели:

A screenshot of a computer program

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%.