Задачи модуль 2

по дисциплине

“Объектно-ориентированное программирование”

Выполнила студентка

группы БФИ1901

Киселева Анна

Москва 2020

**2/6**

1. Создайте функцию, которая повторяет каждый символ в строке n раз.

Пример:

repeat("mice", 5) ➞ "mmmmmiiiiiccccceeeee"

repeat("hello", 3) ➞ "hhheeellllllooo"

repeat("stop", 1) ➞ "stop"

import java.util.Arrays;

public class Task2 {

// 1 программа увечиливает количество каждого символа в n раз

public static String repeat(String word, int n){

String newWord = " ";

for (int i = 0; i< word.length() ; i++){

for (int j = 0; j<n; j++){

newWord+= word.charAt(i);

}

}

return newWord;

}

public static void main(String[] args) {

// Тест 1 - вернет hhheeellllllooo

System.out.println(repeat("hello", 3));

}

1. Создайте функцию, которая принимает массив и возвращает разницу между самыми большими и самыми маленькими числами.

Пример:

differenceMaxMin([10, 4, 1, 4, -10, -50, 32, 21]) ➞ 82

// Smallest number is -50, biggest is 32.

differenceMaxMin([44, 32, 86, 19]) ➞ 67

// Smallest number is 19, biggest is 86.

// 2 находит разницу между минимум и максимумом массива

// минимуму присваивам максимально возможное число

// максимуму - минимально возможное

// находим минимум и максимум введенного массива

// и возвращаем разницу между этими числами

public static int differenceMaxMin(int [] mass){

int max = Integer.MIN\_VALUE;

int min = Integer.MAX\_VALUE;

for (int value : mass) {

min = Math.min(min, value);

max = Math.max(max, value);

}

return max - min;

}

public static void main(String[] args) {

// Тест 2 - вернет 82

System.out.println(differenceMaxMin(new int[]{10, 4, 1, 4, -10, -50, 32, 21}));

}

1. Создайте функцию, которая принимает массив в качестве аргумента и возвращает true или false в зависимости от того, является ли среднее значение всех элементов массива целым числом или нет.

Пример:

isAvgWhole([1, 3]) ➞ true

isAvgWhole([1, 2, 3, 4]) ➞ false

isAvgWhole([1, 5, 6]) ➞ true

isAvgWhole([1, 1, 1]) ➞ true

isAvgWhole([9, 2, 2, 5]) ➞ false

// 3 проверяет, является ли среднее значение всех элементов целым числом

// суммируем все элементы массива

// если среднее арифметическое целое число - возвращаем true

// если нет - false

public static boolean isAvgWhole(int mass[]){

double sum=0;

for (int value : mass) {

sum += value;

}

if ((sum%mass.length)%100<1)

return true;

else

return false;

}

public static void main(String[] args) {

// Тест 3 - вернет false

System.out.println(isAvgWhole(new int[]{1, 2 , 3 , 4}));

}

1. Создайте метод, который берет массив целых чисел и возвращает массив, в котором каждое целое число является суммой самого себя + всех предыдущих чисел в массиве.

Пример:

cumulativeSum([1, 2, 3]) ➞ [1, 3, 6]

cumulativeSum([1, -2, 3]) ➞ [1, -1, 2]

cumulativeSum([3, 3, -2, 408, 3, 3]) ➞ [3, 6, 4, 412, 415, 418]

// 4 возвращает массив

// в котором каждое целое число - сумма самого себя

// и всех предыдущих чисел

public static int[] cumulativeSum( int [] mass){

for (int i = 1; i < mass.length; i++){

mass[i] = mass[i] + mass[i-1];

}

return mass;

}

public static void main(String[] args) {

// Тест 4 - вернет [3, 6, 4, 412, 415, 418]

System.out.println(Arrays.toString(cumulativeSum(new int[]{3, 3, -2, 408, 3, 3})));

}

1. Создайте функцию, которая возвращает число десятичных знаков, которое имеет число (заданное в виде строки). Любые нули после десятичной точки отсчитываются в сторону количества десятичных знаков.

Пример:

getDecimalPlaces("43.20") ➞ 2

getDecimalPlaces("400") ➞ 0

getDecimalPlaces("3.1") ➞ 1

//5 считает число десятичных знаков

// если в числе есть точка

// возвращаем длинну числа - порядковый номер точки - 1

// если точки нет - возвращаем 0

public static int getDecimalPlaces(String number){

if (number.indexOf('.')!=0){

return number.length()-number.indexOf('.')-1;

}

else

return 0;

}

public static void main(String[] args) {

// Тест 5 - вернет 2

System.out.println(getDecimalPlaces("43.20"));

}

1. Создайте функцию, которая при заданном числе возвращает соответствующее число Фибоначчи.

Пример:

Fibonacci(3) ➞ 3

Fibonacci(7) ➞ 21

Fibonacci(12) ➞ 233

//6 возвращается n-ое число Фибоначчи

// если просим вывести 0 или 1 число Фибоначчи,

// то возвращаем 1

public static int Fibonacci (int n){

switch (n) {

case 0:

case 1:

return 1;

default:

return Fibonacci(n - 2) + Fibonacci(n - 1);

}

}

public static void main(String[] args) {

// Тест 6 - вернет 21

System.out.println(Fibonacci(7));

}

1. Почтовые индексы состоят из 5 последовательных цифр. Учитывая строку, напишите функцию, чтобы определить, является ли вход действительным почтовым индексом. Действительный почтовый индекс выглядит следующим образом:

– Должно содержать только цифры (не допускается использование нецифровых цифр).

– Не должно содержать никаких пробелов.

– Длина не должна превышать 5 цифр.

Пример:

isValid("59001") ➞ true

isValid("853a7") ➞ false

isValid("732 32") ➞ false

isValid("393939") ➞ false

// 7 проверяет, правильно ли введен индекс

// если в введенном индексе есть буквы, пробелы или не 5 символов

// то счетчик принимает не нулевое значение

// в противном случае выводится true

public static boolean isValid(String index){

int buf = 0;

for (int i = 0; i < index.length(); i++) {

if (!Character.isDigit(index.charAt(i)))

buf++;

else if (index.charAt(i) == ' ')

buf++;

else if (index.length() != 5)

buf++;

}

return buf == 0;

}

public static void main(String[] args) {

// Тест 7 - вернет true

System.out.println(isValid("15006"));

}

1. Пара строк образует странную пару, если оба из следующих условий истинны:

– Первая буква 1-й строки = последняя буква 2-й строки.

– Последняя буква 1-й строки = первая буква 2-й строки.

– Создайте функцию, которая возвращает true, если пара строк представляет собой странную пару, и false в противном случае.

Пример:

isStrangePair("ratio", "orator") ➞ true

// "ratio" ends with "o" and "orator" starts with "o".

// "ratio" starts with "r" and "orator" ends with "r".

isStrangePair("sparkling", "groups") ➞ true

isStrangePair("bush", "hubris") ➞ false

isStrangePair("", "") ➞ true

//8 странная пара

// если первый символ первого слова и последний второго совпадают

// и первый элемент второго слова совпадает с последним элементом первого

// то вернется true

public static boolean isStrangePair (String word1, String word2){

return word1.charAt(0) == word2.charAt(word2.length() - 1) &&

word2.charAt(0) == word2.charAt(word1.length() - 1);

}

public static void main(String[] args) {

// Тест 8 - вернет false

System.out.println(isStrangePair("bush", "hubris") );

}

1. Создайте две функции: isPrefix(word, prefix-) и isSuffix (word, -suffix).

– isPrefix должен возвращать true, если он начинается с префиксного аргумента.

– isSuffix должен возвращать true, если он заканчивается аргументом суффикса.

– В противном случае верните false.

Пример:

isPrefix("automation", "auto-") ➞ true

isSuffix("arachnophobia", "-phobia") ➞ true

isPrefix("retrospect", "sub-") ➞ false

isSuffix("vocation", "-logy") ➞ false

Примечание:

Аргументы префикса и суффикса имеют тире - в них.

//9 проверяем является ли данный суффикс или префикс частью введенного слова

//убираем из префикса/суффикса -

// проверяем начинается/заканчивается ли введенное слово на суффикс/префикс

public static boolean isPrefix(String word, String pref){

pref = pref.substring(0, pref.length()-1);

//System.out.println(pref);

return word.startsWith(pref);

}

public static boolean isSuffix(String word, String suf){

suf = suf.substring(1);

//System.out.println(suf);

return word.endsWith(suf);

}

public static void main(String[] args) {

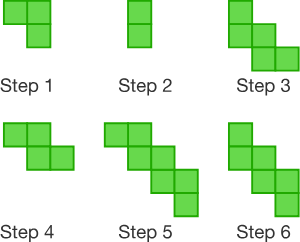
// Тест 9 - вернет true и true

System.out.println(isPrefix("automation", "auto-"));

System.out.println(isSuffix("arachnophobia", "-phobia"));

}

1. Создайте функцию, которая принимает число (шаг) в качестве аргумента и возвращает количество полей на этом шаге последовательности.



Шаг 0: начните с 0

Шаг 1: Добавьте 3

Шаг 2: Вычтите 1

Повторите Шаги 1 И 2 ...

Пример:

boxSeq(0) ➞ 0

boxSeq(1) ➞ 3

boxSeq(2) ➞ 2

// 10 Создайте функцию, которая принимает число (шаг) в качестве аргумента

// и возвращает количество полей на этом шаге последовательности.

// если шаг = 0 , возвращаем 0

// создаем цикл, "делающий" шаги

// если шаг четный, то к счетчику прибавляем 3

// если шаг не четный, то из счетчика вычитается 1

public static int boxSeq(int shag){

if (shag == 0) return 0;

int count = 0;

for (int i = 0; i < shag; i++) {

if (i % 2 != 0) count--;

else count += 3;

}

return count;

}

public static void main(String[] args) {

// Тест 10 - вернет 5

System.out.println(boxSeq(3));

}