

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет: Информатика и вычислительная техника

Кафедра: Кибербезопасность информационных систем

**Отчёт по лабораторным работам**

**по дисциплине “Языки Программирования”**

Выполнил обучающийся гр.ВКБ21

Котелевец Кирилл Александрович

(Фамилия, Имя, Отчество)

Проверила:

Доцент Куликова Ольга Витальевна

(должность, Фамилия, Имя, Отчество)

Ростов-на-Дону

2024

Содержание

[Лабораторная работа №1 3](#_Toc166492945)

[Лабораторная работа № 1.1 21](#_Toc166492946)

[Лабораторная работа № 2 31](#_Toc166492947)

[Лабораторная работа №3 41](#_Toc166492948)

[Лабораторная работа №3.1 50](#_Toc166492949)

[Лабораторная работа №4 55](#_Toc166492950)

# Лабораторная работа №1

**Тема**: Java Массивы и Вычисления

**Цель:** Запрос пароля при открытии или изменении файла

Задание 1

Вычислить где - наибольший элемент массива X(20);  
- наибольший элемент массива Y(15). Для вычисления наибольшего элемента массива использовать функцию.

|  |
| --- |
| package Lab\_1;  public class Task1 {  public static void main(String[] args) {  double z;  int[] x={1,2,3,4,5,6,7,8,9,10,20};  int[] y={1,2,3,4,5,6,7,8,9,10,15};  int max\_x,max\_y;  max\_x = *max*(x);  max\_y = *max*(y);  z = (Math.*exp*(Math.*abs*(max\_x)) - Math.*exp*(Math.*abs*(max\_y))) / Math.*sqrt*((Math.*abs*(max\_x \* max\_y)));  System.*out*.printf("%.10f%n",z);  }  public static int max(int[] array){  int max\_arr = array[0];  for(int i :array) if(i > max\_arr) max\_arr = i;  return max\_arr;  } } |

Задание 2

Даны массивы действительных чисел А(20), В(15), С(10). Вычислить где S, T, K – суммы положительных элементов массивов А, В, С соответственно. Для вычисления суммы положительных элементов использовать функцию.

|  |
| --- |
| package Lab\_1;  public class Task2 {  public static void main(String[] args) {  int[] a = {1,-2,3,4,-5,6,7,8,9,-10,20};  int[] b = {1,2,3,4,-5,-6,7,-8,9,10,15};  int[] c = {1,2,3,-4,5,6,7,-8,9,10,-30};  int s = *sumPositive*(a);  int t = *sumPositive*(b);  int k = *sumPositive*(c);  float m = (float)(s+t+k)/2;  System.*out*.println(m);  }  private static int sumPositive(int[] array){  int sum = 0;  for (int i:array) if(i>0) sum+=i;  return sum;  } } |

Задание 3

Даны целые числа m, n. Вычислить . Для вычисления факториалаиспользовать функцию.

|  |
| --- |
| package Lab\_1;  public class Task3 {  public static void main(String[] args) {  int m = 4,n = 3;  long c = *factorial*(m) / (*factorial*(n) \* *factorial*(m - n));  System.*out*.println(c);  }  private static long factorial(int k) {  long result = 1L;  for (int i = 1; i <= k; i++) result \*= i;  return result;  } } |

Задание 4

Даны действительные х, у, z. Составить программу вычисления значения используя функцию для вычисления выражения

|  |
| --- |
| package Lab\_1;  public class Task4 {  public static void main(String[] args) {  int x = 2, y = 3, z = 4;  double s;  s = *powSqrt*(x, y) + *powSqrt*(x, z) + *powSqrt*(y, z) ;  System.*out*.println(s);  }  private static int pow(int value){  return value \* value;  }   private static double powSqrt(int a, int b) {  return Math.*sqrt*(*pow*(a) + *pow*(b) + Math.*pow*(Math.*sin*(a \* b), 2));  } } |

Задание 5

Составить программу для вычисления среднего арифметического положительных элементов массивов Х(20), Y( 15), Z( 10), используя в качестве подпрограммы функцию.

|  |  |
| --- | --- |
| package Lab\_1;  public class Task5 {  public static void main(String[] args) {  int[] x = {1,-2,3,4,-5,6,7,8,9,-10,20};  int[] y = {1,2,3,4,-5,-6,7,-8,9,10,15};  int[] z = {1,2,3,-4,5,6,7,-8,9,10,-30};  System.*out*.println("Среднее арифметическое положительного массива x = "+ String.*format* ("%.4f", ((double)(*sumPositive*(x))/x.length)));  System.*out*.println("Среднее арифметическое положительного массива y = "+ String.*format* ("%.4f", ((double)(*sumPositive*(y))/y.length)));  System.*out*.println("Среднее арифметическое положительного массива z = "+ String.*format* ("%.4f", ((double)(*sumPositive*(z))/z.length)));  }   private static int sumPositive(int[] array){  int sum = 0;  for (int i:array) if(i>0) sum+=i;  return sum;  } } |  |

Задание 6

Даны массивы A(15), B(10),C(12). Вычислить

Для вычислений использовать функцию.

|  |
| --- |
| package Lab\_1;  public class Task6 {   public static void main(String[] args) {  int[] a = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15};  int[] b = {1, 2, 3, 4, -5, -6, 7, 8, 9, 10, -12};  int[] c = {1, 2, 3, -4, 5, 6, 7, -8, 9, 10, 30, 12};  int l;  if (*min*(a) > 10) l = *min*(b) + *min*(c);  else l=1+*min*(c);  System.*out*.println(l);  }   private static int min(int[] array) {  int min\_arr = array[0];  for (int i : array) if (i < min\_arr) min\_arr = i;  return min\_arr;  } } |

Задание 7

Дан массив D(40) вещественных чисел. Найти среднее геометрическое его элементов, которые удовлетворяют условию 0 <di<12. Для вычислений использовать функцию.

|  |
| --- |
| package Lab\_1;  import java.util.Arrays; import java.util.concurrent.ThreadLocalRandom;  public class Task7 {  public static void main(String[] args) {  double[] D = new double[40];  int len = D.length;  for (int i = 0; i < len; i++) D[i] = ThreadLocalRandom.*current*().nextDouble(20.0);  Arrays.*parallelSort*(D);  int lens = *countRange*(D, len, 0, 12);  System.*out*.println(*meanGeometric*(D,lens));  }  private static int lowerIndex(double[] arr, int len, int x){  int l = 0, mid;  len--;  while (l <= len){  mid = (l + len) / 2;  if (arr[mid] > x) len = mid - 1;  else l = mid + 1;  }  return l;  }  private static int upperIndex(double[] arr, int len, int y){  int l = 0, mid;  len--;  while (l <= len){  mid = (l + len) / 2;  if (arr[mid] < y) l = mid + 1;  else len = mid - 1;  }  return l;  }  private static int countRange(double[] arr, int len, int x, int y){  return *upperIndex*(arr, len, y) - *lowerIndex*(arr, len, x);  }  private static double meanGeometric(double[] arr, int len) {  double sum = 0;  for (int i = 0; i < len; i++)  sum += Math.*log*(arr[i]);  sum /= len;  return Math.*exp*(sum);  } } |

Задание 8

Дан массив А(80) целых чисел. Найти сумму и количество теx элементов массива, которые отрицательны и нечетны. Использовать в качестве подпрограммы процедуру.

|  |
| --- |
| package Lab\_1;  import java.util.Arrays; import java.util.concurrent.ThreadLocalRandom;  public class Task8 {  public static void main(String[] args) {  int[] D = new int[10];  int len = D.length, sum = 0;  for (int i = 0; i < len; i++) D[i] = ThreadLocalRandom.*current*().nextInt(-10,10);  Arrays.*parallelSort*(D);  int posix = *searchNegativeOdd*(D,len);  for (int i = 0; i < posix; i++) if ((D[i] & 1) != 0) sum+=D[i];  System.*out*.println(sum);  }  private static int searchNegativeOdd(int[] arr, int len) {  int l = 0, mid;  len--;  while (l <= len) {  mid = (l + len) / 2;  if (arr[mid] < 0) l = mid + 1;  else len = mid - 1;  }  return l;  } } |

Задание 9

Функция, вычисляющая среднее арифметическое элементов массива. Написать функцию, которая вычисляет среднее арифметическое элементов массива, переданного ей в качестве аргумента.

|  |
| --- |
| package Lab\_1;  import java.util.concurrent.ThreadLocalRandom;  public class Task9 {  public static void main(String[] args) {  int[] arr = new int[10];  int len = arr.length;  for (int i = 0; i < len; i++) arr[i] = ThreadLocalRandom.current().nextInt(100);  System.out.println(meanArithmetic(arr,len));  }  private static double meanArithmetic(int[] arr, int len){  int result = 0;  for (int i:arr) result += i;  return (double)result / len;  } } |

Задание 10

Отсортировать массив по возрастанию суммы цифр. Дан одномерный массив, состоящий из натуральных чисел. Выполнить сортировку данного массива по возрастанию суммы цифр чисел. Например, дан массив чисел [14, 30, 103]. После сортировки он будет таким: [30, 103, 14], так как сумма цифр числа 30 составляет 3, числа 103 равна 4, числа 14 равна 5.

|  |
| --- |
| package Lab\_1; import java.util.Arrays; import java.util.concurrent.ThreadLocalRandom;  public class Task10 {  public static void main(String[] args) {  Integer[] arr = new Integer[10];  int len = arr.length;  for (int i = 0; i < len; i++) arr[i] = ThreadLocalRandom.*current*().nextInt(0,100);  Arrays.*sort*(arr, (Integer x, Integer y) -> {  int tempx = x,tempy = y;  int sumx = 0,sumy = 0;  while(tempx != 0){  sumx += tempx % 10;  tempx /= 10;  }  while(tempy != 0){  sumy += tempy % 10;  tempy /= 10;  }  return sumx - sumy;  });  for(int i:arr) System.*out*.print(i + " ");  }  } |

Задание 11

Вывести на экран исходный массив, отсортированный массив, а также для контроля сумму цифр каждого числа отсортированного массива.

|  |
| --- |
| package Lab\_1;  import java.util.Arrays; import java.util.concurrent.ThreadLocalRandom;  public class Task11 {  public static void main(String[] args) {  Integer[] arr = new Integer[10];  int len = arr.length;  for (int i = 0; i < len; i++) arr[i] = ThreadLocalRandom.current().nextInt(0,100);  System.out.println();  for(int i:arr) System.out.format("%4s",i +" ");  Arrays.sort(arr, (Integer x, Integer y) -> {  int tempx=x,tempy=y;  int sumx=0,sumy=0;  while(tempx!=0){  sumx+=tempx%10;  tempx/=10;  }  while(tempy!=0){  sumy+=tempy%10;  tempy/=10;  }  return sumx-sumy;  });  System.out.println();  for(int i:arr) System.out.format("%4s",i +" ");  System.out.println();  for(int i:arr) System.out.format("%4s", summDigit(i)+" ");  }  private static int summDigit(int temp){  int sum=0;  while(temp!=0){  sum+=temp%10;  temp/=10;  }  return sum;  } } |

Задание 12

Определить количество разрядов числа

Написать функцию, которая определяет количество разрядов введенного целого числа.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task12 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите число:");  int number = keyboard.nextInt();  System.*out*.println(*analyzeRaz*(number));  }  private static int analyzeRaz(int number){  if (number < 100000) {  if (number < 100) {  if (number < 10) {  return 1;  } else {  return 2;  }  } else {  if (number < 1000) {  return 3;  } else {  if (number < 10000) {  return 4;  } else {  return 5;  }  }  }  } else {  if (number < 10000000) {  if (number < 1000000) {  return 6;  } else {  return 7;  }  } else {  if (number < 100000000) {  return 8;  } else {  if (number < 1000000000) {  return 9;  } else {  return 10;  }  }  }  }  } } |

Задание 13

Сумма ряда с факториалом

Вычислить сумму ряда

5

s = ∑ (-1) \* i \* (x / i!)

i=1

Значение x вводится с клавиатуры.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task13 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите число:");  int x = keyboard.nextInt(), s = 0;  for(int i = 0; i<5;i++){  s += (-1) \* i \* (x / *factorial*(i));  }  System.*out*.println(s);  }  private static long factorial(int k) {  long result = 1L;  for (int i = 1; i <= k; i++) result \*= i;  return result;  } } |

Задание 14

Изменить порядок слов в строке на обратный

Вводится строка, состоящая из слов, разделенных пробелами. Следует заменить ее на строку, в которой слова идут в обратном порядке по сравнению с исходной строкой. Вывести измененную строку на экран.

|  |  |
| --- | --- |
| package Lab\_1;  import java.util.Scanner;  public class Task14 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите строку:");  String[] src = keyboard.nextLine().split(" ");  int len = src.length;  String temp;  for (int i = 0; i <= (len / 2);i++){  temp = src[len-i-1];  src[len-i-1] = src[i];  src[i] = temp;  }  System.*out*.println(String.*join*(" ",src));  } } |  |

Задание 15

Функция бинарного поиска в массиве

Пользователь вводит число. Сообщить, есть ли оно в массиве, элементы которого расположены по возрастанию значений, а также, если есть, в каком месте находится. При решении задачи использовать бинарный (двоичный) поиск, который оформить в виде отдельной функции.

|  |
| --- |
| package Lab\_1;  import java.util.Arrays; import java.util.Scanner; import java.util.concurrent.ThreadLocalRandom;  public class Task15 {  public static void main(String[] args) {  int[] arr = new int[10];  int len = arr.length;  for (int i = 0; i < len; i++) arr[i] = ThreadLocalRandom.current().nextInt(0,100);  for(int i:arr) System.out.format("%3s",i +" ");  System.out.println();  Scanner keyboard = new Scanner(System.in);  System.out.println("Введите число:");  int digit\_search = keyboard.nextInt();  Arrays.parallelSort(arr);  int result = binSearch(arr,digit\_search,0,len-1);  System.out.println(result != -1 ? result : "Не найдено");  for(int i:arr) System.out.format("%3s",i +" ");  }  private static int binSearch(int[] sortedArray, int valueToFind, int low, int high){  int index = -1;  while (low <= high) {  int mid = low + (high - low) / 2;  if (sortedArray[mid] < valueToFind) {  low = mid + 1;  } else if (sortedArray[mid] > valueToFind) {  high = mid - 1;  } else if (sortedArray[mid] == valueToFind) {  index = mid;  break;  }  }  return index;  } } |

Задание 16

Вычисление наибольших общих делителей

Найти наибольшие общие делители (НОД) для множества пар чисел.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task16 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.in);  System.out.println("Введите первое число:");  int n1 = keyboard.nextInt();  System.out.println("Введите второе число:");  int n2 = keyboard.nextInt();  System.out.println("НОД: " + algorithmStein(n1,n2));  }  private static int algorithmStein(int n1, int n2) {  if (n1 == 0) return n2;  if (n2 == 0) return n1;  int n;  for (n = 0; ((n1 | n2) & 1) == 0; n++) {  n1 >>= 1;  n2 >>= 1;  }  while ((n1 & 1) == 0) n1 >>= 1;  do {  while ((n2 & 1) == 0) n2 >>= 1;  if (n1 > n2) {  int temp = n1;  n1 = n2;  n2 = temp;  }  n2 = (n2 - n1);  } while (n2 != 0);  return n1 << n;  } } |

Задание 17

Найти площади разных фигур

В зависимости от выбора пользователя вычислить площадь круга, прямоугольника или треугольника. Для вычисления площади каждой фигуры должна быть написана отдельная функция.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task17 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите номер фигуры:");  System.*out*.println("1.Круг");  System.*out*.println("2.Прямоугольник");  System.*out*.println("3.Треугольник");  int param = keyboard.nextInt();  switch (param) {  case 1 -> *circle*();  case 2 -> *rect*();  case 3 -> *triangle*();  }  }  private static void rect() {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите длину:");  int a = keyboard.nextInt();  System.*out*.println("Введите ширину:");  int b = keyboard.nextInt();  System.*out*.println("Площадь прямоугольника: " + a \* b);   }  private static void triangle(){  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите номер данных:");  System.*out*.println("1.Высота и основание");  System.*out*.println("2.Три стороны");  int params = keyboard.nextInt();  double result = switch (params){  case 1 -> {  System.*out*.println("Высота = ");  int h = keyboard.nextInt();  System.*out*.println("Основание = ");  int b = keyboard.nextInt();  yield ((h \* b) / 2.0);  }  case 2 -> {  System.*out*.println("Первая сторона = ");  int a = keyboard.nextInt();  System.*out*.println("Вторая сторона = ");  int b = keyboard.nextInt();  System.*out*.println("Третья сторона = ");  int c = keyboard.nextInt();  if (a < 0 || b < 0 || c < 0 || (a + b <= c) || a + c <= b || b + c <= a){  System.*out*.println("Не верные данные");  System.*exit*(0);  }  double P = (a + b + c) / 2.0;  yield Math.*sqrt*(P \* (P - a) \* (P - b) \* (P - c));   }  default -> {  System.*out*.println("Данный вариант не существует");  System.*exit*(0);  yield 0D;  }   };  System.*out*.println("Площадь треугольника" + result);  }   private static void circle(){  Scanner sc = new Scanner(System.*in*);  System.*out*.println("Введите радиус: ");  double radius = sc.nextDouble();  double S = radius \* radius \* Math.*PI*;  System.*out*.println("Площадь круга" + S);  } } |

Задание 18

Найти массив с максимальной суммой элементов

Сгенерировать десять массивов из случайных чисел. Вывести их и сумму их элементов на экран. Найти среди них один с максимальной суммой элементов. Указать какой он по счету, повторно вывести этот массив и сумму его элементов.

|  |
| --- |
| package Lab\_1;  import java.util.concurrent.ThreadLocalRandom;  public class Task18 {  public static void main(String[] args) {  int len = 10;  int[][] arr = new int[len][len];  for (int i = 0; i < len; i++) {  for (int j = 0; j < len; j++) arr[i][j] = ThreadLocalRandom.current().nextInt(0,100);  }  int[] sum = new int[len];  for (int i =0; i < len; i++) sum[i] = sumArray(arr[i]);  for (int i =0; i < len; i++) {  for (int j = 0; j < len; j++) {  System.out.printf("%4d", arr[i][j]);  }  System.out.printf("%.4s", "| " + sum[i]);  System.out.println();  }  System.out.println("--------------------------------------------");  int max\_sum = searchIndexLargest(sum);  for (int i =0; i < len; i++) System.out.printf("%4d", arr[max\_sum][i]);  System.out.printf("%.4s", "| " + sum[max\_sum]);  }   private static int searchIndexLargest(int[] array){  int index\_largest = 0;  for ( int i = 0; i < array.length; i++ ) {  if ( array[i] > array[index\_largest] ) {  index\_largest = i;  }  }  return index\_largest;  }  private static int sumArray(int[] array){  int temp =0;  for (int i:array) temp+=i;  return temp;  } } |

Задание 19

Заполнение массива и подсчет суммы его элементов оформить в виде отдельной функции.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner; import java.util.concurrent.ThreadLocalRandom;  public class Task19 {  public static void main(String[] args) {  int len = 4;  int[][] arr = new int[len][len];  for (int i = 0; i < len; i++) {  for (int j = 0; j < len; j++) arr[i][j] = ThreadLocalRandom.*current*().nextInt(0,100);  }  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите номер искомой задачи:");  *viewArr*(arr,len);  System.*out*.println("1.Главная диагональ");  System.*out*.println("2.Побочная диагональ");  int param = keyboard.nextInt();  switch (param) {  case 1 -> *diagonalSums*(arr,len,true);  case 2 -> *diagonalSums*(arr,len,false);  default -> System.*out*.println("Не верные данные");  }  }  private static void viewArr(int[][] arr, int len){  for (int i =0; i < len; i++) {  for (int j = 0; j < len; j++) {  System.*out*.printf("%4d", arr[i][j]);  }  System.*out*.println();  }  System.*out*.println("--------------------------------------------");  }  private static void diagonalSums(int [][]mat, int n, boolean f) {  int principal = 0, secondary = 0;  for (int i = 0; i < n; i++) {  principal += mat[i][i];  secondary += mat[i][n - i - 1];  }  System.*out*.println((f) ? ("Главная диагональ:" + principal) : ("Побочная диагональ:" + secondary));  } } |

Задание 20

Вычислить сумму элементов главной или побочной диагонали матрицы

Дана квадратная матрица. Вычислить сумму элементов главной или побочной диагонали в зависимости от выбора пользователя. Сумма элементов любой диагонали должна вычисляться в одной и той же функции.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task20 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите десятичное число:");  int dec = keyboard.nextInt();  *decBinary*(dec);  }  private static void decBinary(int n) {  StringBuilder result = new StringBuilder();  for (int i = 31; i >= 0; i--) {  int k = n >> i;  if ((k & 1) > 0) result.append("1");  else result.append("0");  }  result = new StringBuilder(result.substring(result.indexOf("1"), 32));  System.*out*.println(result);  } } |

Задание 21

Функция перевода десятичного числа в двоичное

Переводить в двоичную систему счисления вводимые в десятичной системе счисления числа до тех пор, пока не будет введен 0. Для перевода десятичного числа в двоичное написать функцию.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task21 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите x:");  int x = keyboard.nextInt(), y;  if (x <= 5){  if(x >= -5) y = *pow*(x);  else y = 2 \* Math.*abs*(x) - 1;  }  else y = 2 \* x;  System.*out*.println("y: " + y);  }  private static int pow(int value){  return value \* value;  } } |

Задание 22

Вычислить значения функции y=f(x) на заданном диапазоне

Вычислить значения нижеприведенной функции в диапазоне значений x от -10 до 10 включительно с шагом, равным 1.

y = x2 при -5 <= x <= 5;

y = 2\*|x|-1 при x < -5;

y = 2x при x > 5.

Вычисление значения функции оформить в виде программной функции, которая принимает значение x, а возвращает полученное значение функции (y).

|  |
| --- |
| package Lab\_1;  import java.util.Scanner; import java.util.concurrent.ThreadLocalRandom;  public class Task22 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите min:");  int min = keyboard.nextInt();  System.*out*.println("Введите max:");  int max = keyboard.nextInt();  int len = 10;  int[] arr = new int[len];  for (int i = 0; i < len; i++) arr[i] = ThreadLocalRandom.*current*().nextInt(min,max);  for (int i = 0; i < len; i++) System.*out*.printf("%.4s",arr[i] + " ");  } } |

Задание 23

Функция заполнения массива случайными числами

Написать функцию, которая заполняет массив случайными числами в диапазоне, указанном пользователем. Функция должна принимать два аргумента - начало диапазона и его конец, при этом ничего не возвращать. Вывод значений элементов массива должен происходить в основной ветке программы.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task23 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите R:");  double R = keyboard.nextDouble();  System.*out*.println("Введите U:");  double U = keyboard.nextDouble();  double I = U / R;  System.*out*.println(I);   } } |

Задание 24

Написать функцию вычисления величины силы тока на участке электрической цепи сопротивлением R Ом при напряжении U В.

Написать функцию вычисления напряжения на каждом из последовательно

соединенных участков электрической цепи сопротивлением R1, R2, R3 Ом, если сила

тока при напряжении U В составляет I А.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task24 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите I:");  double I = keyboard.nextDouble();  double[] R = new double[3], U = new double[3];  for (int i = 0; i < 3;i++){  System.*out*.println("Введите R" + i + ": ");  R[i] = keyboard.nextDouble();  U[i] = I \* R[i];  }  for (int i = 0; i < 3; i++) System.out.printf("%.6s",U[i] + " ");  } } |

Задание 25

Составить программу для ввода на экран номера дня недели и вывода соответствующего ему дня недели на русском языке.

|  |
| --- |
| package Lab\_1;  import java.util.Scanner;  public class Task25 {  public static void main(String[] args) {  Scanner keyboard = new Scanner(System.*in*);  System.*out*.println("Введите номер дня недели:");  int day = keyboard.nextInt();  System.*out*.println(switch(day){  case 1 -> "Понедельник";  case 2 -> "Вторник";  case 3 -> "Среда";  case 4 -> "Четверг";  case 5 -> "Пятница";  case 6 -> "Суббота";  case 7 -> "Воскресенье";  default -> "Покиньте планету";  });  } } |

# Лабораторная работа № 1.1

**Тема:** Разработка простого консольного приложения

Задание 1

Ввести n строк с консоли, найти самую короткую строку. Вывести эту строку и ее длину.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task1 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Введите количество строк: ");  int n = scanner.nextInt();  scanner.nextLine();   String shortestString = null;  int shortestStringLength = Integer.*MAX\_VALUE*;   for (int i = 0; i < n; i++) {  System.*out*.print("Введите строку " + (i + 1) + ": ");  String input = scanner.nextLine();   if (input.length() < shortestStringLength) {  shortestString = input;  shortestStringLength = input.length();  }  }   System.*out*.println("Самая короткая строка: " + shortestString);  System.*out*.println("Длина строки: " + shortestStringLength);  } } |

Задание 2

Ввести n строк с консоли. Упорядочить и вывести строки в порядке возрастания их длин, а также (второй приоритет) значений этих их длин.

|  |
| --- |
| package Lab\_11;  import java.util.Arrays; import java.util.Comparator; import java.util.Scanner;  public class Lab11Task2 {    public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Введите количество строк: ");  int n = scanner.nextInt();  scanner.nextLine(); *// считываем символ новой строки после ввода числа* String[] strings = new String[n];   for (int i = 0; i < n; i++) {  System.*out*.print("Введите строку " + (i + 1) + ": ");  strings[i] = scanner.nextLine();  }   Arrays.*sort*(strings, Comparator.*comparingInt*(String::length).thenComparing(String::compareTo));   System.*out*.println("Отсортированные строки: ");  for (String str : strings) {  System.*out*.println(str);  System.*out*.println("Длина строки: " + str.length());  }  } } |

Задание 3

Ввести n строк с консоли. Вывести на консоль те строки, длина которых меньше средней, также их длины.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task3 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.print("Введите количество строк: ");  int n = scanner.nextInt();   scanner.nextLine(); *// считываем перевод на новую строку после ввода числа* String[] strings = new String[n];  double sumLength = 0;   System.*out*.println("Введите строки:");  for (int i = 0; i < n; i++) {  strings[i] = scanner.nextLine();  sumLength += strings[i].length();  }   double averageLength = sumLength / n;   System.*out*.println("Строки, длина которых меньше средней:");  for (String str : strings) {  if (str.length() < averageLength) {  System.*out*.println(str + " (длина: " + str.length() + ")");  }  }  }  } |

Задание 4

В каждом слове текста k-ю букву заменить заданным символом. Если k больше длины слова, корректировку не выполнять.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task4 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   System.*out*.println("Введите текст:");  String text = scanner.nextLine();   System.*out*.println("Введите номер буквы, которую нужно заменить:");  int k = scanner.nextInt();   System.*out*.println("Введите символ для замены:");  char symbol = scanner.next().charAt(0);   String[] words = text.split(" ");   for (String word : words) {  if (k <= word.length()) {  char[] letters = word.toCharArray();  letters[k - 1] = symbol;  System.*out*.print(new String(letters) + " ");  } else {  System.*out*.print(word + " ");  }  }  } } |

Задание 5

В русском тексте каждую букву заменить ее номером в алфавите. В одной строке печатать текст с двумя пробелами между буквами, в следующей строке внизу под каждой буквой печатать ее номер.

|  |
| --- |
| package Lab\_11; import java.util.Scanner;  public class Lab11Task5 {  public static void main(String[] args) {  String text = "Пример нашего текста";  text = text.toLowerCase(); *// Преобразуем текст к нижнему регистру* for (int i = 0; i < text.length(); i++) {  char letter = text.charAt(i);  int number = letter - 'а' + 1; *// Вычисляем номер буквы от 1 до 33 для русского алфавита* System.out.print(letter + " ");  }  System.out.println(); *// Переход на новую строку* for (int i = 0; i < text.length(); i++) {  char letter = text.charAt(i);  int number = letter - 'а' + 1; *// Вычисляем номер буквы от 1 до 33 для русского алфавита* if (letter == ' '){System.out.print(" ");}  else {System.out.print(number + " ");}   }  } } |

Задание 6

Из небольшого текста удалить все символы, кроме пробелов, не являющиеся буквами. Между последовательностями подряд идущих букв оставить хотя бы один пробел.

|  |
| --- |
| package Lab\_11;  public class Lab11Task6 {  public static void main(String[] args) {  String text = "Пример 123 текста 456! Символы удалить.";  String cleanedText = text.replaceAll("[^а-яА-Яa-zA-Z ]+", " "); *// Удаление символов, не являющихся буквами или пробелами* cleanedText = cleanedText.replaceAll("(\\p{L}+)(\\s\*\\1)+", "$1 "); *// Оставляем только один пробел между последовательностями букв* System.out.println(cleanedText);  } } |

Задание 7

Из текста удалить все слова заданной длины, начинающиеся на согласную букву.

|  |
| --- |
| import java.util.regex.Matcher; import java.util.regex.Pattern;  public class Lab11Task7 {  public static void main(String[] args) {  String text = "Example text to illustrate the removal of words of length 5 that start with a consonant.";   int wordLength = 5;  String consonantPattern = "[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]";   Pattern pattern = Pattern.*compile*("\\b" + consonantPattern + "\\p{L}{" + (wordLength - 1) + "}\\b");  Matcher matcher = pattern.matcher(text);   String cleanedText = matcher.replaceAll("");   System.*out*.println(cleanedText);  } } |

Задание 8

В тексте найти все пары слов, из которых одно является об­ращением другого.

|  |
| --- |
| package Lab\_11;  public class Lab11Task8 {  public static void main(String[] args) {  String text = "Просто здравствуй просто как дела алед как ты привет отсорП ";  System.*out*.print("Исходный текст: " + text);  String[] words = text.split(" ");  StringBuilder reverseWord;  for (int i = 0; i < words.length; i++) {  for (int j = i + 1; j < words.length; j++) {  reverseWord = new StringBuilder(words[j]).reverse();  if (words[i].equalsIgnoreCase(reverseWord.toString())) {  System.*out*.println("\n" + words[i] + " - " + reverseWord.reverse());    }    }  }  } } |

Задание 9

Найти и напечатать, сколько раз повторяется в тексте каждое слово.

|  |
| --- |
| package Lab\_11;  public class Lab11Task9 {  public static void main(String[] args) {  String text = "А снег идёт, а снег идёт и всё мерцает и плывёт плывёт";  System.*out*.println("Исходный текст: " + text);  String[] words = text.split(" ");  for (int i = 0; i < words.length; i++) {  int repeat = 0;  for (int j = 0; j < words.length; j++) {  if (words[i].equals(words[j])) {  repeat++;  }  }  System.*out*.println("Слово " + "'" + words[i] + "'" + " повторяется " + repeat + " раз(а)");  }    }  } |

Задание 10

Найти, каких букв, гласных или согласных, больше в каждом предложении текста.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task10 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  System.*out*.println("Введите текст:");  String text = scanner.nextLine();   String[] sentences = text.split("[.!?]");   for (String sentence : sentences) {  int vowels = 0;  int consonants = 0;   for (char c : sentence.toLowerCase().toCharArray()) {  if (c >= 'a' && c <= 'z') {  if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {  vowels++;  } else {  consonants++;  }  }  }   System.out.println("В предложении: \"" + sentence.trim() + "\"");  if (vowels > consonants) {  System.out.println("Больше гласных букв");  } else if (consonants > vowels) {  System.out.println("Больше согласных букв");  } else {  System.out.println("Гласных и согласных букв поровну");  }  }   scanner.close();  } }  *//Programming is interesting and exciting. Get started right now!* |

Задание 11

Выбрать три разные точки заданного на плоскости множества точек, составляющие треугольник наибольшего периметра.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task11 {  public static void main(String[] args) {  Scanner in = new Scanner(System.*in*);  System.*out*.print("Введите количество точек: ");  int n = in.nextInt();  double[] x\_kord = new double[n];  double[] y\_kord = new double[n];  for (int i = 0; i < n; i++) {  System.out.print("Введите количество координату x точки номер " + (i + 1) + ": ");  x\_kord[i] = in.nextDouble();  System.out.print("Введите количество координату y точки номер " + (i + 1) + ": ");  y\_kord[i] = in.nextDouble();  }  System.out.println("Вы ввели точки: ");  for (int i = 0; i < n; i++) {  System.out.println("(" + x\_kord[i] + ";" + y\_kord[i] + ")");  }   if (n < 3) {  System.out.print("Точек недостаточно чтобы построить треугольник");  } else {  if (n == 3) {  double res = lenght(0, 1, x\_kord, y\_kord) + lenght(1, 2, x\_kord, y\_kord) + lenght(2, 0, x\_kord, y\_kord);  System.out.print("Единственный и наибольший периметр: " + res);  } else {  double max = 0;  double[] x\_max = new double[3];  double[] y\_max = new double[3];  for (int i = 0; i < n; i++) {  for (int j = i; j < n; j++) {  for (int k = j; k < n; k++) {  double maybe = lenght(i, j, x\_kord, y\_kord) + lenght(j, k, x\_kord, y\_kord) + lenght(k, i, x\_kord, y\_kord);  if (maybe >= max) {  max = maybe;  y\_max[0] = y\_kord[i];  y\_max[1] = y\_kord[j];  y\_max[2] = y\_kord[k];  }  }  }  }  System.out.println("Максимальный периметр треугольника " + max);  for (int i = 0; i < 3; i++) {  System.out.println("(" + x\_kord[i] + ";" + y\_kord[i] + ")");  }  }  }  }   private static double lenght(int i, int j, double[] x, double[] y) {  double dif\_x = x[j] - x[i];  double dif\_y = y[j] - y[i];  double ans = Math.sqrt(Math.pow(dif\_x, 2) + Math.pow(dif\_y, 2));  System.out.println(ans);  return ans;  } } |

Задание 12

Найти такую точку заданного на плоскости множества точек, сумма расстояний от которой до остальных минимальна.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task12 {  public static void main(String[] args) {  Scanner in = new Scanner(System.*in*);  System.*out*.print("Введите количество точек: ");  int n = in.nextInt();  double[] x\_kord = new double[n];  double[] y\_kord = new double[n];  for (int i = 0; i < n; i++) {  System.*out*.print("Введите количество координату x точки номер " + (i + 1) + ": ");  x\_kord[i] = in.nextDouble();  System.*out*.print("Введите количество координату y точки номер " + (i + 1) + ": ");  y\_kord[i] = in.nextDouble();  }  System.*out*.println("Вы ввели точки:");  for (int i = 0; i < n; i++) {  System.*out*.println("(" + x\_kord[i] + ";" + y\_kord[i] + ")");  }  double max = 0;  double x\_max = 0;  double y\_max = 0;  int max\_index = 0;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  double sum = 0;  if (j != i) {  sum += *lenght*(i, j, x\_kord, y\_kord);  }  if (sum >= max) {  max = sum;  x\_max = x\_kord[i];  y\_max = y\_kord[i];  max\_index = i + 1;  }  }  }  System.*out*.println("Ответ: точка номер " + max\_index + " с координатами (" + x\_max + ";" + y\_max + ") и суммой расстояний " + max);  }   private static double lenght(int i, int j, double[] x, double[] y) {  double dif\_x = x[j] - x[i];  double dif\_y = y[j] - y[i];  double ans = Math.*sqrt*(Math.*pow*(dif\_x, 2) + Math.*pow*(dif\_y, 2));  return ans;  } } |

Задание 13

Выпуклый многоугольник задан на плоскости перечислением координат вершин в порядке обхода его границы. Определить площадь многоугольника.

|  |
| --- |
| package Lab\_11;  import java.util.Scanner;  public class Lab11Task13 {  public static void main(String[] args){  Scanner in = new Scanner(System.*in*);  System.*out*.print("Введите количество точек: ");  int n = in.nextInt();  int[] x = new int[n];  int[] y = new int[n];  for (int i = 0; i < n; i++){  System.*out*.print("Введите количество координату x точки номер "+ (i + 1) +": ");  x[i] = in.nextInt();  System.*out*.print("Введите количество координату y точки номер "+ (i + 1) +": ");  y[i] = in.nextInt();  }  System.*out*.println("Вы ввели точки:");  for (int i = 0; i < n; i++) {  System.*out*.println("(" + x[i] + ";" + y[i] + ")");  }  double s ;  int smid = 0;  for (int i = 0; i < n-1; i++){  if (i + 1 == n) {  System.*out*.println(i);  smid += (x[i] \* y[1]) - (y[i] \* x[1]);  }  else{  smid += (x[i] \* y[i+1]) - (y[i] \* x[i+1]);  }  }  System.*out*.println(smid);  s = Math.*abs*(smid) / 2;  System.*out*.print("Площадь многоугольников равна " + s);  } } |

# Лабораторная работа № 2

**Тема:** преобразование типов в Java

Задание 1

Расставьте правильно операторы приведения типа, чтобы получился ответ: d = 3.765. Операторы — в условии.

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  int a = 15;  int b = 4;  float c = a / b;  double d = a \* 1e-3 + c;  System.out.println(d);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную a типа int.

Метод main() должен содержать переменную b типа int.

Метод main() должен содержать переменную c типа float.

Метод main() должен содержать переменную d типа double.

Начальное значение переменных при инициализации менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить число 3.765.

Задание 2

Давайте тоже найдем решение задачи: у нас есть какие-то переменные, преобразованные в другой тип, но их недостаточно. Нужно добавить одну операцию по преобразованию типа, чтобы получался нужный нам ответ b = 0.

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  float f = (float) 128.50;  int i = (int) f;  int b = (int) (i + f);  System.out.println(b);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную f типа float.

Метод main() должен содержать переменную i типа int.

Метод main() должен содержать переменную b типа int.

Начальное значение переменных при инициализации менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить число 0.

Задание 3

Даны short number = 9, char zero = ‘0’ и int nine = (zero + number).

Добавьте одну операцию по преобразованию типа, чтобы получился красивый правильный ответ: 9.

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  short number = 9;  char zero = '0';  int nine = (zero + number);  System.out.println(nine);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран. В ней можно только добавлять операторы приведения типа.

Метод main() должен содержать переменную number типа short.

Метод main() должен содержать переменную zero типа char.

Метод main() должен содержать переменную nine типа int.

Начальное значение переменных при инициализации менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить число 9.

Задание 4

Уберите ненужные операторы приведения типа, чтобы получился ответ: result: 1000.0

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  double d = (short) 2.50256e2d;  char c = (short) 'd';  short s = (short) 2.22;  int i = 150000;  float f = 0.50f;  double result = f + (i / c) - (d \* s) - 500e-3;  System.out.println("result: " + result);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную c типа char.

Метод main() должен содержать переменную s типа short.

Метод main() должен содержать переменную i типа int.

Метод main() должен содержать переменную f типа float.

Метод main() должен содержать переменную d типа double.

Метод main() должен содержать переменную result типа double.

Начальное значение переменных при инициализации менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить текст "result: 1000.0".

Задание 5

Уберите ненужные операторы приведения типа, чтобы получился ответ: 1234567.

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  long l = (byte) 1234\_564\_890L;  int x = (byte) 0b1000\_1100\_1010;  double m = (byte) 110\_987\_654\_6299.123\_34;  float f = (byte) l++ + 10 + ++x - (float) m;  l = (long) f / 1000;  System.out.println(l);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную l типа long.

Метод main() должен содержать переменную x типа int.

Метод main() должен содержать переменную m типа double.

Метод main() должен содержать переменную f типа float.

Значения переменных менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить текст 1234567.

Задание 6

Нужно добавить одну операцию по преобразованию типа, чтобы получался ответ: d = 2.941. Пример вывода: 2.9411764705882355

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  int a = 50;  int b = 17;  double d = a / b;  System.out.println(d);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную a типа int.

Метод main() должен содержать переменную b типа int.

Метод main() должен содержать переменную d типа double.

Значения переменных менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить число 2.9411764705882355.

Задание 7

Нужно добавить одну операцию по преобразованию типа, чтобы получался ответ: d = 1.0

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  int a = 257;  int b = 4;  int c = 3;  int e = 2;  double d = a + b / c / e;  System.out.println(d);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную a типа int.

Метод main() должен содержать переменную b типа int.

Метод main() должен содержать переменную c типа int.

Метод main() должен содержать переменную e типа int.

Метод main() должен содержать переменную d типа double.

Значения переменных менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить число 1.0.

Задание 8

Вам надо добавить одну операцию по преобразованию типа, чтобы получался ответ: d = 5.5.

|  |
| --- |
| public class Solution {  public static void main(String[] args) {  int a = 5;  int b = 4;  int c = 3;  int e = 2;  double d = a + b / c / e;  System.out.println(d);  }  } |

Требования:

Программа должна выводить текст на экран.

Нельзя менять команду вывода на экран.

Метод main() должен содержать переменную a типа int.

Метод main() должен содержать переменную b типа int.

Метод main() должен содержать переменную c типа int.

Метод main() должен содержать переменную e типа int.

Метод main() должен содержать переменную d типа double.

Значения переменных менять нельзя. Можно добавлять только операторы приведения типа.

Программа должна выводить число 5.5.

Решение:

Листинг Программы

|  |
| --- |
| */\*  \* Класс TaskExecutorLabTwo содержит методы для выполнения различных задач из лабораторной работы No2.  \* В основном это обычный вывод текста в поле. \*/* package com.kikuzawa.laboratoriesjava.Labs;  public class TaskExecutorLabTwo {  private static String Task1() {  return """  public class Lab2Task1 {  public static void main(String[] args) {  int a = 15;  int b = 4;  float c = (float) a / b;  double d = a \* 1e-3 + c;  System.out.println(d);  }  }""";  }   private static String Task2() {  return """  public class Lab2Task2 {  public static void main(String[] args) {  float f = (float) 128.50;  int i = (int) f;  int b = (byte) (int) (i + f);  System.out.println(b);  }  }""";  }   private static String Task3() {  return """  public static void main(String[] args) {  short number = 9;  char zero = '0';  int nine = (Character.getNumericValue(zero) + number);  System.out.println(nine);  }""";  }   private static String Task6() {  return """  public class Lab2Task4 {  public static void main(String[] args) {  double d = (short) 2.50256e2d;  char c = 'd';  short s = (short) 2.22;  int i = 150000;  float f = 0.50f;  double result = f + (i / c) - (d \* s) - 500e-3;  System.out.println("result: " + result);  }  }""";  }   private static String Task4() {  return """  public class Lab2Task5 {  public static void main(String[] args) {  long l = 1234\_564\_890L;  int x = 0b1000\_1100\_1010; //бинарность  double m = (byte) 110\_987\_654\_6299.123\_34;  float f = l++ + 10 + ++x - (float) m;  l = (long) f / 1000;  System.out.println(l);  }  }  """;  }   private static String Task5() {  return """  public class Lab2Task6 {  public static void main(String[] args) {  int a = 50;  int b = 17;  double d = (double) a / b;  System.out.println(d);  }  }  """;  }   private static String Task7() {  return """  public class Lab2Task7 {  public static void main(String[] args) {  int a = 257;  int b = 4;  int c = 3;  int e = 2;  double d = (byte) a + b / c / e;  System.out.println(d);  }  }""";  }   private static String Task8() {  return """  public class Lab2Task8 {  public static void main(String[] args) {  int a = 5;  int b = 4;  int c = 3;  int e = 2;  double d = a + b / c / (double) e;  System.out.println(d);  }  }""";  }   public static String execute(int task) {  return switch (task) {  case 1 -> *Task1*();  case 2 -> *Task2*();  case 3 -> *Task3*();  case 4 -> *Task4*();  case 5 -> *Task5*();  case 6 -> *Task6*();  case 7 -> *Task7*();  case 8 -> *Task8*();  default -> "Пока не реализовано";  };  } } |

# Лабораторная работа №3

**Тема:** Строки в Java

**Задания**

Часть 1.

1. Ввести n строк с консоли, найти самую короткую и самую длинную строки. Вывести найденные строки и их длину.

2. Ввести n строк с консоли. Упорядочить и вывести строки в порядке возрастания (убывания) значений их длины.

3. Ввести n строк с консоли. Вывести на консоль те строки, длина которых меньше (больше) средней, а также длину.

4. Ввести n слов с консоли. Найти слово, в котором число различных символов минимально. Если таких слов несколько, найти первое из них.

5. Ввести n слов с консоли. Найти количество слов, содержащих только символы латинского алфавита, а среди них – количество слов с равным числом гласных и согласных букв.

6. Ввести n слов с консоли. Найти слово, символы в котором идут в стро-гом порядке возрастания их кодов. Если таких слов несколько, найти первое из них.

7. Ввести n слов с консоли. Найти слово, состоящее только из различных символов. Если таких слов несколько, найти первое из них.

8. Ввести n слов с консоли. Среди слов, состоящих только из цифр, найти слово-палиндром. Если таких слов больше одного, найти второе из них.

9. Написать программы решения задач 1–8, осуществляя ввод строк как аргументов командной строки.

Часть 2.

1. a) Напишите метод, который принимает в качестве параметра любую строку, например “I like Java!!!”.

б) Распечатать последний символ строки. Используем метод String.charAt().

в) Проверить, заканчивается ли ваша строка подстрокой “!!!”. Используем метод String.endsWith().

г) Проверить, начинается ли ваша строка подстрокой “I like”. Используем метод String.startsWith().

д) Проверить, содержит ли ваша строка подстроку “Java”. Используем метод String.contains().

e) Найти позицию подстроки “Java” в строке “I like Java!!!”.

ж) Заменить все символы “а” на “о”.

з) Преобразуйте строку к верхнему регистру.

и) Преобразуйте строку к нижнему регистру.

к) Вырезать строку Java c помощью метода String.substring().

2. а) Дано два числа, например 3 и 56, необходимо составить следующие строки:

3 + 56 = 59

3 – 56 = -53

3 \* 56 = 168.

Используем метод StringBuilder.append().

б) Замените символ “=” на слово “равно”. Используйте методы StringBuilder.insert(), StringBuilder.deleteCharAt().

в) Замените символ “=” на слово “равно”. Используйте методы StringBuilder.replace().

3. Напишите метод, заменяющий в строке каждое второе вхождение «object-oriented programming» (не учитываем регистр символов) на «OOP». Например, строка

"Object-oriented programming is a programming language model organized around objects rather than "actions" and data rather than logic. Object-oriented programming blabla. Object-oriented programming bla."

должна быть преобразована в

"Object-oriented programming is a programming language model organized around objects rather than "actions" and data rather than logic. OOP blabla. Object-oriented programming bla."

4. Даны строки разной длины (длина - четное число), необходимо вернуть ее два средних знака: "string" → "ri", "code" → "od", "Practice"→"ct".

5. Создать строку, используя форматирование: Студент [Фамилия] получил [оценка] по [предмету]. Форматирование и вывод строки на консоль написать в отдельном методе, который принимает фамилию, оценку и название предмета в качестве параметров. Выделить под фамилию 15 символов, под оценку 3 символа, предмет – 10.

6. Дана строка “Versions: Java 5, Java 6, Java 7, Java 8, Java 12.” Найти все подстроки "Java X" и распечатать их.

7. Найти слово, в котором число различных символов минимально. Слово может содержать буквы и цифры. Если таких слов несколько, найти первое из них. Например, в строке "fffff ab f 1234 jkjk" найденное слово должно быть "fffff".

8. Предложение состоит из нескольких слов, разделенных пробелами. Например: "One two three раз два три one1 two2 123 ". Найти количество слов, содержащих только символы латинского алфавита.

9. Предложение состоит из нескольких слов, например: "Если есть хвосты по дз, начните с 1 не сданного задания. 123 324 111 4554". Среди слов, состоящих только из цифр, найти слово палиндром.

**Решение**

Листинг Программы

|  |
| --- |
| */\*\*  \* Класс TaskExecutorLabThree выполняет различные задачи из лабораторной 3 над строками на Java.  \* Задачи включают поиск самых длинных и коротких строк, сортировку по длине,  \* работу со словами и символами и другие операции.  \*/* package com.kikuzawa.laboratoriesjava.Labs;  import java.util.Arrays; import java.util.regex.Matcher; import java.util.regex.Pattern; import java.util.List;  import static com.kikuzawa.laboratoriesjava.classes.HelpModule.\*;  public class TaskExecutorLabThree {   private static String Task1(String input) {  String[] strings = input.split(" ");  String result = "";  result += "Самая длинная строка: " + *max*(strings) + "\n" + "Ее длина:" + *max*(strings).length();  result += "\nСамая короткая строка: " + *min*(strings) + "\n" + "Ее длина:" + *min*(strings).length();   return result;  }   private static String Task2(String input) {  String[] strings = input.split(" ");  String result = "";   result += "Исходный список:\n" + String.*join*("\n", strings) + "\n";  result += "По возрастанию:\n" + String.*join*("\n", *sortByLength*(Arrays.*asList*(strings))) + "\n";  result += "По убыванию:\n" + String.*join*("\n", *sortByLengthAnti*(Arrays.*asList*(strings)));   return result;  }   private static String Task3(String input) {  String[] strings = input.split(" ");  StringBuilder result;  double average = *AverageLength*(Arrays.*asList*(strings));   result = new StringBuilder("Исходный список:\n" + String.*join*("\n", strings) + "\n Больше среднего:\n");  for (String word : *filtStringsUpNumber*(strings, average)) {  result.append(word).append(word.length()).append("\n");  }  result.append("Меньше среднего:\n");  for (String word : *filtStringsDownNumber*(strings, average)) {  result.append(word).append(word.length()).append("\n");  }  return result.toString();  }   private static String Task4(String input) {  String[] words = input.split(" ");   return *findWordWithMinDistinctChars*(words);  }   private static String Task5(String input) {  String[] words = input.split(" ");   String result = "";   int numLatinWords = *countLatinWords*(words);  int numLatinWordsWithEqualVowelsAndConsonants = *countLatinWordsWithEqualVowelsAndConsonants*(words);   result += "Количество слов, содержащих только символы латинского алфавита: " + numLatinWords + "\n";  result += "Количество слов с равным числом гласных и согласных букв: " + numLatinWordsWithEqualVowelsAndConsonants;   return result;  }   private static String Task6(String input) {  String[] words = input.split(" ");   String ascendingWord = *findAscendingWord*(words);   *// Выведите результат* if (ascendingWord != null) {  return String.*format*("Слово с символами в строгом порядке возрастания их кодов: " + ascendingWord);  } else {  return "Таких слов нет.";  }  }   private static String Task7(String input) {  String[] words = input.split(" ");  String uniqueWord = *findUniqueWord*(words);   if (uniqueWord != null) {  return String.*format*("Слово, состоящее только из различных символов: " + uniqueWord);  } else {  return "Таких слов нет.";  }  }   private static String Task8(String input) {  String[] words = input.split(" ");   *// Проверьте, все ли слова состоят из цифр* boolean allDigits = true;  for (String word : words) {  if (!word.matches("[0-9]+")) {  allDigits = false;  break;  }  }   String DigitPalindrome = null;  if (allDigits) {  DigitPalindrome = *findSecondDigitPalindrome*(words);  }    if (DigitPalindrome != null) {  return String.*format*("Слово-палиндром, состоящее только из цифр: " + DigitPalindrome);  } else if (!allDigits) {  return "Есть слова, состоящие не только из цифр!";  } else {  return "Таких слов нет!";  }  }   private static String Task10(String input) {  String result = "a) " + input + "\n";   int lastIndex = input.length() - 1;  char lastChar = input.charAt(lastIndex);   result += "б) " + lastChar + "\n";   boolean endsWith = input.endsWith("!!!");  result += "в) " + endsWith + "\n";   boolean startsWith = input.startsWith("I like");  result += "г) " + startsWith + "\n";   boolean contains = input.contains("Java");  result += "д) " + contains + "\n";   int index = input.indexOf("Java");  result += "е) " + index + "\n";   String newStr = input.replace('a', 'o');  result += "ж) " + newStr + "\n";   String upperCase = input.toUpperCase();  result += "з) " + upperCase + "\n";   String lowerCase = input.toLowerCase();  result += "и) " + lowerCase + "\n";   String substring = input.substring(7, 11); *// вырезает подстроку с индексов 7 по 10 (не включая 11)* result += "к) " + substring + "\n";   return result;   }   private static String Task11(String input) {  String[] nums = input.split(" ");  boolean flag = true;   for (String num : nums) {  if (!num.matches("[0-9]+")) {  flag = false;  break;  }  }   if (nums.length != 2) {  flag = false;  }    if (flag) {  String result = "";  StringBuilder sb = new StringBuilder();  int num1 = Integer.*parseInt*(nums[0]);  int num2 = Integer.*parseInt*(nums[1]);   sb.append(num1).append(" + ").append(num2).append(" = ").append(num1 + num2).append("\n");  sb.append(num1).append(" - ").append(num2).append(" = ").append(num1 - num2).append("\n");  sb.append(num1).append(" \* ").append(num2).append(" = ").append(num1 \* num2);   result += "а)\n" + sb + "\n";    int index = sb.indexOf("=");  while (index != -1) {  sb.deleteCharAt(index);  sb.insert(index, "равно");  index = sb.indexOf("=", index + 1);  }   result += "б)\n" + sb + "\n";   *//sb.replace(sb.indexOf("="), sb.indexOf("=") + 1, "равно");* result += "в)\n" + sb + "\n";   return result;   } else {  return "Неправильный ввод!";  }  }   private static String Task12(String input) {  String target = "Object-oriented programming";  String replacement = "OOP";  return *replaceEverySecondOccurrence*(input, target, replacement);  }   private static String Task13(String input) {  String[] strings = input.split(" ");  StringBuilder result1 = new StringBuilder();  StringBuilder result2 = new StringBuilder();  for (String str : strings) {  if (*checkWordLength*(str)) {  String result = *getMiddleCharacters*(str);  result1.append("Слово \"").append(str).append("\" прошло проверку на длину. Результат: ").append(result).append("\n");  } else {  result2.append("Слово \"").append(str).append("\" не прошло проверку на длину.\n");  }  }   return result1 + "------------\n" + result2;  }   private static String Task14(String input) {  String[] words = input.split(" ");   try {  if (words.length != 3) {  return "Неправильный ввод!";  } else {  String name = words[0];  int grade = Integer.*parseInt*(words[1]);  String subject = words[2];   return String.*format*("Студент %15s получил %3d по %10s\n", name, grade, subject);  }  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task15(String input) {   Pattern pattern = Pattern.*compile*("Java\\s+[0-9]+");  StringBuilder result = new StringBuilder();   *// Создание объекта Matcher для поиска совпадений в строке input* Matcher matcher = pattern.matcher(input);   *// Поиск всех совпадений и печать найденных подстрок* while (matcher.find()) {  result.append(matcher.group()).append("\n");  }   return result.toString();  }   private static String Task16(String input) {  String[] words = input.split(" ");    String minUniqueWord = "";  int minUniqueChars = Integer.*MAX\_VALUE*;   for (String word : words) {  int uniqueChars = *countUniqueCharacters*(word);  if (uniqueChars < minUniqueChars) {  minUniqueChars = uniqueChars;  minUniqueWord = word;  }  }   return "Слово с минимальным числом различных символов: " + minUniqueWord;  }   private static String Task17(String input) {  String[] words = input.split(" ");   String result = Integer.*toString*(*countLatinWords*(words));   return "Количество таких слов: " + result;  }   private static String Task18(String input){  List<String> words = Arrays.*asList*(input.split(" "));   List<String> numericWords = words.stream()  .filter(word -> word.matches("[0-9]+"))  .toList();   *// Найти все палиндромы* List<String> palindromes = numericWords.stream()  .filter(word -> word.contentEquals(new StringBuilder(word).reverse()))  .toList();    *// Вывести результат* return String.*valueOf*(palindromes);   }   public static String execute(String input, int task) {  return switch (task) {  case 1 -> *Task1*(input);  case 2 -> *Task2*(input);  case 3 -> *Task3*(input);  case 4 -> *Task4*(input);  case 5 -> *Task5*(input);  case 6 -> *Task6*(input);  case 7 -> *Task7*(input);  case 8 -> *Task8*(input);  case 10 -> *Task10*(input);  case 11 -> *Task11*(input);  case 12 -> *Task12*(input);  case 13 -> *Task13*(input);  case 14 -> *Task14*(input);  case 15 -> *Task15*(input);  case 16 -> *Task16*(input);  case 17 -> *Task17*(input);  case 18 -> *Task18*(input);  default -> "Пока не реализовано";  };  } } |

# Лабораторная работа №3.1

**Тема:** Обработка строк. Регулярные выражения

**Задания**

1. Написать регулярное выражение, определяющее является ли данная строка строкой "abcdefghijklmnopqrstuv18340" или нет.

– пример правильных выражений: abcdefghijklmnopqrstuv18340.

– пример неправильных выражений: abcdefghijklmnoasdfasdpqrstuv18340.

2. Написать регулярное выражение, определяющее является ли данная строка GUID с или без скобок. Где GUID это строчка, состоящая из 8, 4, 4, 4, 12 шестнадцатеричных цифр разделенных тире.

– пример правильных выражений: e02fd0e4-00fd-090A-ca30-0d00a0038ba0.

– пример неправильных выражений: e02fd0e400fd090Aca300d00a0038ba0.

3. Написать регулярное выражение, определяющее является ли заданная строка правильным MAC-адресом.

– пример правильных выражений: aE:dC:cA:56:76:54.

– пример неправильных выражений: 01:23:45:67:89:Az.

4. Написать регулярное выражение, определяющее является ли данная строчка валидным URL адресом. В данной задаче правильным URL считаются адреса http и https, явное указание протокола также может отсутствовать. Учитываются только адреса, состоящие из символов, т.е. IP адреса в качестве URL не присутствуют при проверке. Допускаются поддомены, указание порта доступа через двоеточие, GET запросы с передачей параметров, доступ к подпапкам на домене, допускается наличие якоря через решетку. Однобуквенные домены считаются запрещенными. Запрещены спецсимволы, например «–» в начале и конце имени домена. Запрещен символ «\_» и пробел в имени домена. При составлении регулярного выражения ориентируйтесь на список правильных и неправильных выражений заданных ниже.

– пример правильных выражений: http://www.example.com, http://example.com.

– пример неправильных выражений: Just Text, http://a.com.

5. Написать регулярное выражение, определяющее является ли данная строчка шестнадцатиричным идентификатором цвета в HTML. Где #FFFFFF для белого, #000000 для черного, #FF0000 для красного и т.д.

– пример правильных выражений: #FFFFFF, #FF3421, #00ff00.

– пример неправильных выражений: 232323, f#fddee, #fd2.

6. Написать регулярное выражение, определяющее является ли данная строчка датой в формате dd/mm/yyyy. Начиная с 1600 года до 9999 года.

– пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003.

– пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899.

7. Написать регулярное выражение, определяющее является ли данная строчка валидным E-mail адресом согласно RFC под номером 2822.

– пример правильных выражений: user@example.com, root@localhost

– пример неправильных выражений: bug@@@com.ru, @val.ru, Just Text2.

8. Составить регулярное выражение, определяющее является ли заданная строка IP адресом, записанным в десятичном виде.

– пример правильных выражений: 127.0.0.1, 255.255.255.0.

– пример неправильных выражений: 1300.6.7.8, abc.def.gha.bcd.

9. Проверить, надежно ли составлен пароль. Пароль считается надежным, если он состоит из 8 или более символов. Где символом может быть английская буква, цифра и знак подчеркивания. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву и одну цифру.

– пример правильных выражений: C00l\_Pass, SupperPas1.

– пример неправильных выражений: Cool\_pass, C00l.

10. Проверить является ли заданная строка шестизначным числом, записанным в десятичной системе счисления без нулей в старших разрядах.

– пример правильных выражений: 123456, 234567.

– пример неправильных выражений: 1234567, 12345.

11. Есть текст со списками цен. Извлечь из него цены в USD, RUR, EU.

– пример правильных выражений: 23.78 USD.

– пример неправильных выражений: 22 UDD, 0.002 USD.

12. Проверить существуют ли в тексте цифры, за которыми не стоит «+».

– пример правильных выражений: (3 + 5) – 9 × 4.

– пример неправильных выражений: 2 \* 9 – 6 × 5.

13. Создать запрос для вывода только правильно написанных выражений со скобками (количество открытых и закрытых скобок должно быть одинаково).

– пример правильных выражений: (3 + 5) – 9 × 4.

– пример неправильных выражений: ((3 + 5) – 9 × 4.

**Решение**

Листинг Программы

|  |
| --- |
| */\*\*  \* Класс TaskExecutorLabThreeDThree предоставляет метод execute для выполнения различных задач из лабораторной работы 3.1  \* по регулярным выражениям. Задачи включают проверку соответствия ввода определенным шаблонам, таким как  \* URL, IP-адреса, электронные адреса и т. д.  \*/* package com.kikuzawa.laboratoriesjava.Labs;  import java.util.regex.Matcher; import java.util.regex.Pattern;  @SuppressWarnings("Annotator") public class TaskExecutorLabThreeDThree {  public static String execute(String input, int task) {  return switch (task) {  case 1 -> *Task1*(input);  case 2 -> *Task2*(input);  case 3 -> *Task3*(input);  case 4 -> *Task4*(input);  case 5 -> *Task5*(input);  case 6 -> *Task6*(input);  case 7 -> *Task7*(input);  case 8 -> *Task8*(input);  case 9 -> *Task9*(input);  case 10 -> *Task10*(input);  case 11 -> *Task11*(input);  case 12 -> *Task12*(input);  case 13 -> *Task13*(input);  default -> "Пока не реализовано";  };  }   private static String Task1(String input) {  String pattern = "^abcdefghijklmnopqrstuv18340";  return String.*valueOf*(Pattern.*matches*(pattern, input));  }   private static String Task2(String input) {  String pattern = "^[A-Za-z0-9]{8}-[A-Za-z0-9]{4}-[A-Za-z0-9]{4}-[A-Za-z0-9]{4}-[A-Za-z0-9]{12}$";  return String.*valueOf*(Pattern.*matches*(pattern, input));  }   private static String Task3(String input) {  String pattern = "^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$";  return String.*valueOf*(Pattern.*matches*(pattern, input));  }   private static String Task4(String text) {  final String URL\_REGEX = "((http|https)://)(www.)?" +  "[a-zA-Z0-9@:%.\\+~#?&//=]" +  "{2,256}\\.[a-z]" +  "{2,6}\\b([-a-zA-Z0-9@:%" +  ".\\+~#?&//=]\*)";  final String IP\_ADDRESS\_REGEX = "\\b(?:\\d{1,3}\\.){3}\\d{1,3}\\b";   Pattern urlPattern = Pattern.compile(URL\_REGEX);  Pattern ipAddressPattern = Pattern.compile(IP\_ADDRESS\_REGEX);   if (ipAddressPattern.matcher(text).find()) {  return "false";  } else {  return String.valueOf(urlPattern.matcher(text).matches());  }  }   private static String Task5(String text){  Pattern pattern = Pattern.compile("^#[A-Fa-f0-9]{6}$");  return String.valueOf(pattern.matcher(text).matches());  }   private static String Task6(String text){  String pattern = "^(0[1-9]|1\\d|2[0-8])/(0[1-9]|1[0-2])/((?:1[6-9]|[2-9]\\d)?\\d{2})\\|29/02/(?:(?:(?:1[6−9]∣[2−9]\\d)(?:0[′ r′48]∣[2468][048]∣[13579][26]))∣(?:16∣[2468][048]∣[3579][26])00)\\|29/02/(?:(?:(?:1[6−9]∣[2−9]\\d)(?:0[′ r′48]∣[2468][048]∣[13579][26]))∣(?:16∣[2468][048]∣[3579][26])00)";  return String.valueOf(Pattern.matches(pattern, text));  }   private static String Task7(String text){  String pattern = "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$";  return String.valueOf(Pattern.matches(pattern, text));  }   private static String Task8(String text){  String pattern = "^([0-9]{1,3}[.]){3}[0-9]{1,3}$";  return String.valueOf(Pattern.matches(pattern, text.strip()));  }   private static String Task9(String text){  String pattern = "^[a-zA-Z0-9\_]{8,}";  if (Pattern.matches(pattern, text) && text.matches(".[A-Z].") && text.matches(".[a-z].") && text.matches(".\\d.")) {  return "True";  } else {  return "False";  }  }  private static String Task10(String text){  Pattern pattern = Pattern.compile("^[1-9]{6}");  return String.valueOf(pattern.matcher(text).matches());  } private static String Task11(String text){  String pattern = "(?:^|[\\n\\r]|[^\\w\\d\\.])([1-9]\\d\*(?:.\\d{,2})?\\s\*(?:USD|EU|RUR))\b";   Pattern r = Pattern.compile(pattern);  Matcher m = r.matcher(text);   StringBuilder result = new StringBuilder();  while (m.find()) {  result.append(m.group(1)).append("\n");  }   return result.toString();  }  private static String Task12(String text){  String pattern = "\b\\d+(?:\\s\*\\+)";   Pattern r = Pattern.compile(pattern);  Matcher m = r.matcher(text.strip());   boolean found = m.find();   return Boolean.toString(found);  }  private static String Task13(String text){  Pattern pattern = Pattern.compile("^[^()]\*$|^[^()]\*(\\([^()]\*\\)[^()]\*)\*$");  Matcher match = pattern.matcher(text.strip());  return String.valueOf(match.matches() && match.group(0).chars().filter(c -> c == '(').count() == match.group(0).chars().filter(c -> c == ')').count());  } } |

# Лабораторная работа №4

**Тема:** Списки в Java

**Задания**

**Односвязные списки**

1. Инициализация списка
2. Добавление элемента в начало списка
3. Добавление элемента в конец списка
4. Показ всех элементов списка
5. Удаление всех элементов списка
6. Определение количества элементов списка
7. Проверка списка на пустоту
8. Удаление первого элемента
9. Удаление последнего элемента
10. Поиск данного значения в списке
11. Поиск наибольшего и наименьшего значений в списке
12. Удаление элемента списка с данным значением
13. Удаление всех элементов списка с данным значением
14. Изменение всех элементов списка с данным значением на новое.
15. Определение, является ли список симметричным.
16. Определение, можно ли удалить из списка каких-нибудь два элемента так, чтобы новый список оказался упорядоченным.
17. Определение, сколько различных значений содержится в списке.
18. Удаление из списка элементов, значения которых уже встречались в предыдущих элементах.
19. Изменение порядка элементов на обратный.
20. Сортировка элементов списка двумя способами (изменение указателей, изменение значений элементов)
21. Двусвязный (двунаправленный) список. Список функций аналогичен предыдущей задаче.
22. Дан упорядоченный список книг. Добавить новую книгу, сохранив упорядоченность списка.
23. Даны два упорядоченных по возрастанию списка. Объедините их в новый упорядоченный по возрастанию список.
24. Дан список целых чисел. Упорядочьте по возрастанию только: а) положительные числа; б) элементы с четными порядковыми номерами в списке.
25. Даны два списка. Определите, совпадают ли множества их элементов.
26. Дан список. После каждого элемента добавьте предшествующую ему часть списка.
27. Пусть элементы списка хранят символы предложения. Замените каждое вхождение слова "itmathrepetitor" на "silence".
28. Дан текстовый файл. Создайте двусвязный список, каждый элемент которого содержит количество символов в соответствующей строке текста.
29. Создайте двусвязный список групп факультета. Каждая группа представляет собой односвязный список студентов.
30. Дан список студентов. Элемент списка содержит фамилию, имя, отчество, год рождения, курс, номер группы, оценки по пяти предметам. Упорядочите студентов по курсу, причем студенты одного курса располагались в алфавитном порядке. Найдите средний балл каждой группы по каждому предмету. Определите самого старшего студента и самого младшего студентов. Для каждой группы найдите лучшего с точки зрения успеваемости студента.

**Решение**

Листинг Программы

|  |
| --- |
| */\*\*  \* Класс TaskExecutorLabFour содержит реализации различных задач из лабораторной работы 4,  \* связанных с операциями над списками и работой со студентами и книгами.  \* Включает методы для выполнения конкретных задач, таких как инициализация списка,  \* добавление элементов, удаление, поиск, сортировка и другие операции.  \*/* package com.kikuzawa.laboratoriesjava.Labs;  import com.kikuzawa.laboratoriesjava.classes.\*;  import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; import java.util.\*; import java.util.stream.Collectors; import java.util.stream.IntStream;   @SuppressWarnings({"CallToPrintStackTrace", "DuplicatedCode"}) public class TaskExecutorLabFour {   static final LinkedListOperations *list1* = new LinkedListOperations();  static final DoublyLinkedList *list2* = new DoublyLinkedList();  static final List<Book> *books* = new ArrayList<>();   static final List<Student1> *students* = new ArrayList<>();   public static String execute(String input, int task) {  return switch (task) {  case 1 -> *Task1From20*(input);  case 2 -> *Task2From20*(input);  case 3 -> *Task3*(input);  case 4 -> *Task4*(input);  case 5 -> *Task5*(input);  case 6 -> *Task6*(input);  case 7 -> *Task7*(input);  case 8 -> *Task8*();  case 9 -> *Task9*();  case 10 -> *Task10*();  case 11 -> *Task11*(input);  default -> "Пока не реализовано";  };  }   private static String Task1From20(String input1) {  try {  *//noinspection DuplicatedCode* String[] numberStrings = input1.split(" ");  String task = numberStrings[0];  StringJoiner joiner = new StringJoiner(" ");   for (int i = 1; i < numberStrings.length; i++) {  joiner.add(numberStrings[i]);  }   String input = joiner.toString();   return switch (task) {  case "1\*" -> *Task1\_1*(input);  case "2\*" -> *Task1\_2*(input);  case "3\*" -> *Task1\_3*(input);  case "4\*" -> *Task1\_4*();  case "5\*" -> *Task1\_5*();  case "6\*" -> *Task1\_6*();  case "7\*" -> *Task1\_7*();  case "8\*" -> *Task1\_8*();  case "9\*" -> *Task1\_9*();  case "10\*" -> *Task1\_10*(input);  case "11\*" -> *Task1\_11*();  case "12\*" -> *Task1\_12*(input);  case "13\*" -> *Task1\_13*(input);  case "14\*" -> *Task1\_14*(input);  case "15\*" -> *Task1\_15*();  case "16\*" -> *Task1\_16*();  case "17\*" -> *Task1\_17*();  case "18\*" -> *Task1\_18*();  case "19\*" -> *Task1\_19*();  case "20\*" -> *Task1\_20*();  default -> throw new IllegalStateException("Unexpected value: " + task);  };    } catch (Exception e) {  throw new RuntimeException(e);  }    }   private static String Task1\_1(String input) {  try {  *list1*.deleteAllElements();  *list1*.initializeList(input);  return "Инизиализация нового списка: " + *list1*.printList();  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_2(String input) {  try {  String oldResult = list1.printList();  int num = Integer.parseInt(input);  list1.addElementToStart(num);  return String.format("Изначальный список: %s \n Добавление элемента %s \n Новый список: %s", oldResult, num, list1.printList());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_3(String input) {  try {  String oldResult = list1.printList();  int num = Integer.parseInt(input);  list1.addElementToEnd(num);  return String.format("Изначальный список: %s \n Добавление элемента %s \n Новый список: %s", oldResult, num, list1.printList());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_4() {  return list1.printList();  }   private static String Task1\_5() {  try {  String oldResult = list1.printList();  list1.deleteAllElements();  return String.format("Изначальный список: %s \n Успешно очистен!", oldResult);  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_6() {  return "В списке:\n" + list1.printList() + "\nЭлементов - " + list1.countElements();  }   private static String Task1\_7() {  return "Проверка списка на пустоту.\nРезультат: " + list1.isListEmpty();  }   private static String Task1\_8() {  try {  String oldResult = list1.printList();  list1.deleteFirstElement();  return String.format("Изначальный список: %s \nУдаление первого элемента\nНовый список: %s", oldResult, list1.printList());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_9() {  try {  String oldResult = list1.printList();  list1.deleteLastElement();  return String.format("Изначальный список: %s \nУдаление последнего элемента\nНовый список: %s", oldResult, list1.printList());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_10(String input) {  try {  int num = Integer.parseInt(input);  return String.format("Список: %s \nПринадлежность элемента %s к списку - %s", list1.printList(), num, list1.searchElement(num));  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task1\_11() {  return String.format("Список %s\nНаибольшее значение - %s\nНаименьшее значение - %s", list1.printList(), list1.findMaximumValue(), list1.findMinimumValue());  }   private static String Task1\_12(String input) {  try {  String oldResult = list1.printList();  int num = Integer.parseInt(input);  list1.deleteElement(num);  return String.format("Исходный список %s\nУдаление элемента со значением %s\nНовый список %s", oldResult, num, list1.printList());  } catch (NumberFormatException e) {  throw new RuntimeException(e);  }  }   private static String Task1\_13(String input) {  try {  String oldResult = list1.printList();  int num = Integer.parseInt(input);  list1.deleteAllElementsWithValue(num);  return String.format("Исходный список %s\nУдаление всех элементов со значением %s\nНовый список %s", oldResult, num, list1.printList());  } catch (NumberFormatException e) {  throw new RuntimeException(e);  }  }   private static String Task1\_14(String input) {  try {  String oldResult = list1.printList();  String[] srt = input.split(" ");  int num1 = Integer.parseInt(srt[0]);  int num2 = Integer.parseInt(srt[1]);  list1.replaceElement(num1, num2);  return String.format("Список: %s\nЗамена элемента %s на %s\nНовый список: %s", oldResult, num1, num2, list1.printList());   } catch (NumberFormatException e) {  throw new RuntimeException(e);  }  }   private static String Task1\_15() {  return "Симметричность списка " + list1.printList() + " - " + list1.isListSymmetric();  }   private static String Task1\_16() {  return "Результат проверки списка " + list1.printList() + " - " + list1.canRemoveTwoElementsToOrderedList();  }   private static String Task1\_17() {  return "Различных значений содержится в списке: " + list1.countDistinctValues();  }   private static String Task1\_18() {  String oldResult = list1.printList();  list1.deleteDuplicateElements();  return "Старый список " + oldResult + "\nНовый список " + list1.printList();  }   private static String Task1\_19() {  String oldResult = list1.printList();  list1.reverseListOrder();  return "Старый список " + oldResult + "\nНовый список " + list1.printList();  }   *//TODO* private static String Task1\_20() {  String oldResult = list1.ListToString();   DoublyLinkedList list = new DoublyLinkedList();  list.clear();  String[] numberStrings = oldResult.split(" ");   for (String numberString : numberStrings) {  int number = Integer.parseInt(numberString);  list.addToEnd(number);  }  DoublyLinkedList listK = new DoublyLinkedList();  listK.clone(list);   list.sortLinkedListByPointers();  listK.sortLinkedListByValues();    return "Старый список " + oldResult + "\nНовый список с изменением указателей " + list.display() + "\nНовый список с изменением значения " + listK.display();   }   private static String Task2From20(String input1) {  try {  *//noinspection DuplicatedCode* String[] numberStrings = input1.split(" ");  String task = numberStrings[0];  StringJoiner joiner = new StringJoiner(" ");   for (int i = 1; i < numberStrings.length; i++) {  joiner.add(numberStrings[i]);  }    String input = joiner.toString();   return switch (task) {  case "1\*" -> Task2\_1(input);  case "2\*" -> Task2\_2(input);  case "3\*" -> Task2\_3(input);  case "4\*" -> Task2\_4();  case "5\*" -> Task2\_5();  case "6\*" -> Task2\_6();  case "7\*" -> Task2\_7();  case "8\*" -> Task2\_8();  case "9\*" -> Task2\_9();  case "10\*" -> Task2\_10(input);  case "11\*" -> Task2\_11();  case "12\*" -> Task2\_12(input);  case "13\*" -> Task2\_13(input);  case "14\*" -> Task2\_14(input);  case "15\*" -> Task2\_15();  case "16\*" -> Task2\_16();  case "17\*" -> Task2\_17();  case "18\*" -> Task2\_18();  case "19\*" -> Task2\_19();  case "20\*" -> Task2\_20();  default -> throw new IllegalStateException("Unexpected value: " + task);  };    } catch (Exception e) {  throw new RuntimeException(e);  }  }   private static String Task2\_1(String input) {  try {  list2.clear();  String[] numberStrings = input.split(" ");   for (String numberString : numberStrings) {  int number = Integer.parseInt(numberString);  list2.addToEnd(number);  }    return "Инизиализация нового списка: " + list2.display();  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_2(String input) {  try {  String oldResult = list2.display();  int num = Integer.parseInt(input);  list2.addToFront(num);  return String.format("Изначальный список: %s \n Добавление элемента %s \n Новый список: %s", oldResult, num, list2.display());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_3(String input) {  try {  String oldResult = list2.display();  int num = Integer.parseInt(input);  list2.addToEnd(num);  return String.format("Изначальный список: %s \n Добавление элемента %s \n Новый список: %s", oldResult, num, list2.display());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_4() {  return list2.display();  }   private static String Task2\_5() {  try {  String oldResult = list2.display();  list2.clear();  return String.format("Изначальный список: %s \n Успешно очистен!", oldResult);  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_6() {  return "В списке:\n" + list2.display() + "\nЭлементов - " + list2.getSize();  }   private static String Task2\_7() {  return "Проверка списка на пустоту.\nРезультат: " + list2.isEmpty();  }   private static String Task2\_8() {  try {  String oldResult = list2.display();  list2.removeFirst();  return String.format("Изначальный список: %s \nУдаление первого элемента\nНовый список: %s", oldResult, list2.display());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_9() {  try {  String oldResult = list2.display();  list2.removeLast();  return String.format("Изначальный список: %s \nУдаление последнего элемента\nНовый список: %s", oldResult, list2.display());  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_10(String input) {  try {  int num = Integer.parseInt(input);  return String.format("Список: %s \nПринадлежность элемента %s к списку - %s", list2.display(), num, list2.contains(num));  } catch (Exception e) {  return "Ошибка: " + e;  }  }   private static String Task2\_11() {  return String.format("Список %s\nНаибольшее значение - %s\nНаименьшее значение - %s", list2.display(), list2.findMax(), list2.findMin());  }   private static String Task2\_12(String input) {  try {  String oldResult = list2.display();  int num = Integer.parseInt(input);  list2.remove(num);  return String.format("Исходный список %s\nУдаление элемента со значением %s\nНовый список %s", oldResult, num, list2.display());  } catch (NumberFormatException e) {  throw new RuntimeException(e);  }  }   private static String Task2\_13(String input) {  try {  String oldResult = list2.display();  int num = Integer.parseInt(input);  list2.removeAll(num);  return String.format("Исходный список %s\nУдаление всех элементов со значением %s\nНовый список %s", oldResult, num, list2.display());  } catch (NumberFormatException e) {  throw new RuntimeException(e);  }  }   private static String Task2\_14(String input) {  try {  String oldResult = list2.display();  String[] srt = input.split(" ");  int num1 = Integer.parseInt(srt[0]);  int num2 = Integer.parseInt(srt[1]);  list2.replace(num1, num2);  return String.format("Список: %s\nЗамена элемента %s на %s\nНовый список: %s", oldResult, num1, num2, list2.display());   } catch (NumberFormatException e) {  throw new RuntimeException(e);  }  }   private static String Task2\_15() {  return "Симметричность списка " + list2.display() + " - " + list2.isSymmetric();  }   private static String Task2\_16() {  return "Результат проверки списка " + list2.display() + " - " + list2.canRemoveTwoElementsToMakeSorted();  }   private static String Task2\_17() {  return "Различных значений содержится в списке: " + list2.countUniqueValues();  }   private static String Task2\_18() {  String oldResult = list2.display();  list2.removeDuplicates();  return "Старый список " + oldResult + "\nНовый список " + list2.display();  }   private static String Task2\_19() {  String oldResult = list2.display();  list2.reverse();  return "Старый список " + oldResult + "\nНовый список " + list2.display();  }   private static String Task2\_20() {  DoublyLinkedList list2\_2 = new DoublyLinkedList();  list2\_2.clone(list2);  String oldResult = list2.display();  list2.sortLinkedListByPointers();  list2\_2.sortLinkedListByValues();  return "Исходный список: " + oldResult + "\nСортировка путем изменения указателей: " + list2.display() + "\n// Сортировка путем изменения значений элементов " + list2\_2.display();  }    private static String Task3(String input) {  if (input != null) {  books.add(new Book(input));  }   StringBuilder result = new StringBuilder();  for (Book element : books) {  result.append(element.title()).append("\n");  }   return result.toString();  }   private static String Task4(String input) {  try {  *// [1, 3, 5, 7] [2, 4, 6, 8, 11]* var it = Arrays.stream(input.split("]\\s+\\[")).iterator();  var firstList = ListMerger.parser(it.next());  var secondList = ListMerger.parser(it.next());   return String.format("Результат слияния: %s", ListMerger.mergeSortedLists(firstList, secondList));   } catch (NoSuchElementException e) {  return "Вы ввели неправильные данные";  }  }   private static String Task5(String input) {  var linkedList = ListMerger.parser(input.substring(input.indexOf("[")));   if (input.startsWith("положительные числа")) {   linkedList.sort((o1, o2) -> {  if (o1 > 0 && o2 > 0)  return o1.compareTo(o2);   if (o1 > 0)  return -1;   if (o2 > 0)  return 1;   return 0;  });   return "Результат сортировки: " + linkedList;    } else {   var sortedEvenIndex = IntStream.range(0, linkedList.size())  .filter(i -> i % 2 == 0)  .mapToObj(linkedList::get)  .sorted()  .iterator();   return "После сортировки: " + IntStream.range(0, linkedList.size())  .mapToObj(index -> {  if (index % 2 == 0) {  return sortedEvenIndex.next();  }  return linkedList.get(index);  }).toList();   }  }   private static String Task6(String input) {  try {  var it = Arrays.stream(input.split("]\\s+\\[")).iterator();  var firstList = new HashSet<>(ListMerger.parser(it.next()));  var secondList = new HashSet<>(ListMerger.parser(it.next()));   return firstList.equals(secondList) ? "Множества элементов совпадают" : "Множества элементов не совпадают";   } catch (NoSuchElementException e) {  return "Вы ввели неправильные данные";  }  }   private static String Task7(String input) {  var past = new StringBuilder();  var list = new ArrayList<>(List.of(input.split("\\s+")));   return list.stream().map(x -> {  var result = past + x;  past.append(x);  return result;  }).collect(Collectors.joining(" "));  }   private static String Task8() {  ArrayList<Character> sentenceList = new ArrayList<>(Arrays.asList(  'i', 't', 'm', 'a', 't', 'h', 'r', 'e', 'p', 'e', 't', 'i', 't', 'o', 'r',  ' ', ' ', 'p', 'r', 'i', 'v', 'e', 't'  ));   var sentence = String.join("", sentenceList.stream().map(Object::toString).toArray(String[]::new));   return sentence.replaceAll("itmathrepetitor", "silence");  }   private static String Task9() {  LinkedList<Integer> charCountList = new LinkedList<>();   try (BufferedReader reader = new BufferedReader(new FileReader("src/main/java/com/kikuzawa/kiku/texts/lab4Task9.txt"))) {  String line;  while ((line = reader.readLine()) != null) {  charCountList.add(line.length());  }  } catch (IOException e) {  e.printStackTrace();  }   StringBuilder result = new StringBuilder("Количество символов в каждой строке:\n");  for (Integer count : charCountList) {  result.append(count).append(" ");  }   return result.toString();  }   private static String Task10() {  LinkedList<Group> facultyGroups = new LinkedList<>();   Group group1 = new Group("ВКБ21");  group1.addStudent(new Student("Котелевец Кирилл"));  group1.addStudent(new Student("Заболотный Иван"));   Group group2 = new Group("ВКБ22");  group2.addStudent(new Student("Ковалев Данил"));  group2.addStudent(new Student("Карпов Роман"));   facultyGroups.add(group1);  facultyGroups.add(group2);   StringBuilder result = new StringBuilder("Группы факультета:\n");  for (Group group : facultyGroups) {  result.append(group).append("\n");  }  return result.toString();  }   private static String Task11(String input) {   String[] words = input.split(";");  List<Integer> grades = new ArrayList<>();  String[] graddes = words[6].split(" ");  for (String i : graddes) {  grades.add(Integer.parseInt(i));  }   students.add(new Student1(words[0], words[1], words[2], Integer.parseInt(words[3]), Integer.parseInt(words[4]), words[5], grades));    *// Сортировка студентов по курсу и алфавиту* students.sort(Comparator.comparingInt(Student1::getCourse).thenComparing(Student1::toString));   *// Нахождение среднего балла каждой группы по каждому предмету* Map<String, Map<String, Double>> groupAverageGrades = new HashMap<>();  for (Student1 student : students) {  if (!groupAverageGrades.containsKey(student.groupNumber)) {  groupAverageGrades.put(student.groupNumber, new HashMap<>());  }  for (int i = 0; i < student.grades.size(); i++) {  String subject = "Предмет " + (i + 1);  double averageGrade = groupAverageGrades.get(student.groupNumber).getOrDefault(subject, 0.0);  averageGrade = (averageGrade \* students.size() + student.grades.get(i)) / (students.size() + 1);  groupAverageGrades.get(student.groupNumber).put(subject, averageGrade);  }  }   *// Нахождение самого старшего и младшего студентов* Student1 oldestStudent = students.stream().min(Comparator.comparingInt(Student1::getBirthYear)).orElse(null);  Student1 youngestStudent = students.stream().max(Comparator.comparingInt(Student1::getBirthYear)).orElse(null);   *// Нахождение лучшего студента в каждой группе* Map<String, Student1> bestStudentsByGroup = new HashMap<>();  for (Student1 student : students) {  if (!bestStudentsByGroup.containsKey(student.groupNumber) ||  student.getAverageGrade() > bestStudentsByGroup.get(student.groupNumber).getAverageGrade()) {  bestStudentsByGroup.put(student.groupNumber, student);  }  }   StringBuilder result = new StringBuilder();  result.append("Средняя успеваемость по группе:\n");  for (Map.Entry<String, Map<String, Double>> entry : groupAverageGrades.entrySet()) {  result.append("Группа: ").append(entry.getKey()).append("\n");  for (Map.Entry<String, Double> gradeEntry : entry.getValue().entrySet()) {  result.append(gradeEntry.getKey()).append(": ").append(gradeEntry.getValue()).append("\n");  }  }   result.append("Старший студент: ").append(oldestStudent).append("\n");  result.append("Молодой студент: ").append(youngestStudent).append("\n");   result.append("Лучшие студенты по группе:\n");  for (Map.Entry<String, Student1> entry : bestStudentsByGroup.entrySet()) {  result.append("Группа: ").append(entry.getKey()).append(", Лучший студент: ").append(entry.getValue()).append("\n");  }  return result.toString();  } } |