|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| **Институт информационных технологий** |
| **Кафедра корпоративных информационных систем**  **Лабораторная работа №2**  по дисциплине  Объектно-ориентированное программирование  **Тема лабораторной работы:** Строки в Java. Применение регулярных выражений для поиска заданных слов.   |  |  | | --- | --- | | Студент группы ИКБО-08-18 | Валяев Данила Андреевич | |  |  | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) | |  |  | |  |  | | Работа представлена к защите | «20» октября 2019 г. | |  |  | |  |  | |
|  |
| **Содержание**  [**Задание 1.** 3](#_Toc27907352)  [**Задание 2** 5](#_Toc27907353)  [**Задание 3** 8](#_Toc27907353)  [**Заключение** 10](#_Toc27907356)  [**Список используемых источников** 11](#_Toc27907357)  [**Приложение** 11](#_Toc27907358) |

# **Задание 1.** Создать приложение - основную программу, которое должно иметь интерфейс для демонстрации работы всех трех заданий (два последних пункта будут реализовываться по мере отладки предыдущих). В каждом пункте создаются свои объекты, для решения задач задания.

Строка представляет собой последовательность символов. Для работы со строками в Java определен класс String, который предоставляет ряд методов для манипуляции строками. Физически объект String представляет собой ссылку на область в памяти, в которой размещены символы.

Для создания новой строки мы можем использовать один из конструкторов класса String, либо напрямую присвоить строку в двойных кавычках

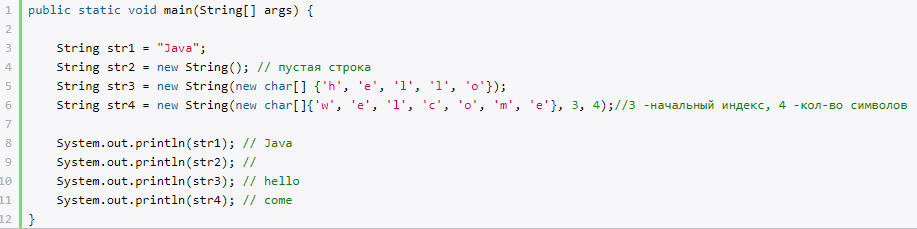


Рисунок 1.1 – Создание строки

В соответствии с вариантом, необходимо:

1. Определить, сколько абзацев в тексте.

Реализация алгоритма определения количества абзацев:

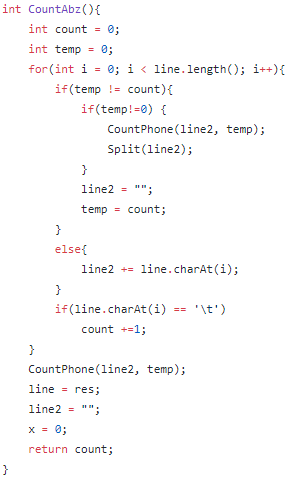


Рисунок 1.2 – определения количества абзацев

1. Определить, суммарное время разговоров в каждом абзаце и сохранить в массиве.

Реализация алгоритма определения суммарного времени разговора:

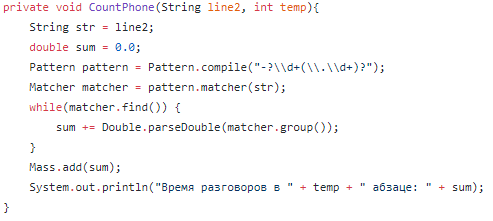


Рисунок 1.3 – определения суммарного времени разговора

1. Разделить первый абзац на столько абзацев, сколько предложений

в абзаце.

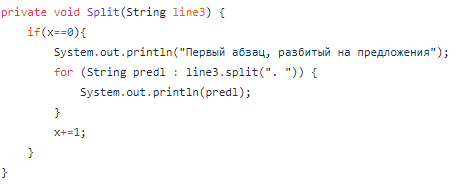


Рисунок 1.4 – разделение на абзацы

**Задание 2.** Реализовать второй пункт консольного интерфейса приложения,

созданного в задании 1, включив алгоритм демонстрации выполнения

задач варианта.

В соответствии с вариантом, необходимо:

1. Удалить самый длинный абзац.

Реализация алгоритма удаления самого длинного абзаца:

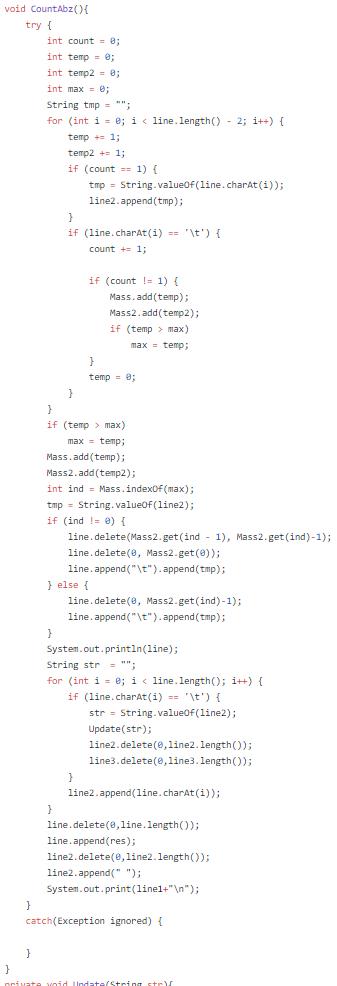


Рисунок 2.1 – удаление самого длинного абзаца

1. Дробные числа, представленные в форме с фиксированной точкой, заменить на числа с плавающей точкой.

Реализация алгоритма преобразования чисел:

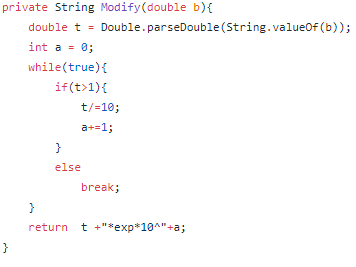


Рисунок 2.2 – преобразование чисел

1. Выполнить модификацию текста, первый абзац перенести в конец текста.

Реализация алгоритма модификации текста:

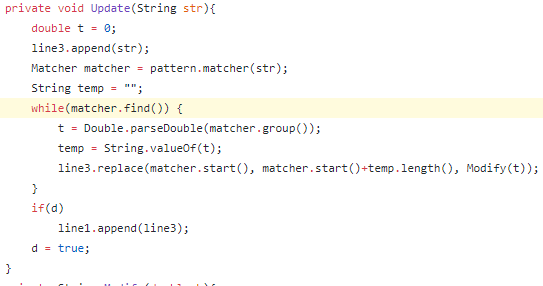


Рисунок 2.3 – модификация текста

# **Задание 3.** Задачи на регулярные выражения.

Регулярные выражения — язык для построения шаблонов поиска и осуществления манипуляций с строками в тексте. Позволяет выполнять проверку строк на соответствие правилам шаблона, разбивать строки, производить поиск и замены отдельных подстрок.

В соответствии с вариантом, необходимо:

1. Поверить соответствие строки формату IP-адреса v4.

Реализация алгоритма проверки соответствия IPv4

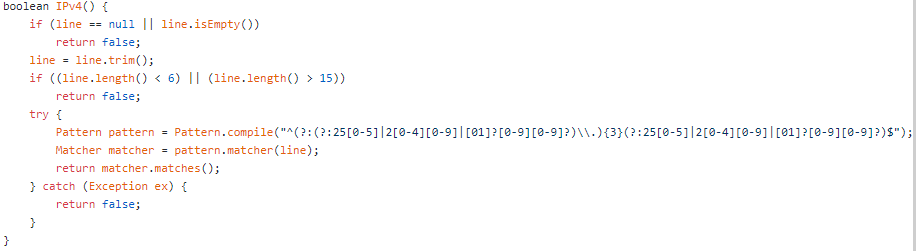


Рисунок 3.1 – алгоритм проверки соответствия IPv4

1. В тексте найти все строки со значением IP-адреса v6 и его элементы на числа в двоичной системе счисления.

# Реализация алгоритма проверки соответствия IPv6



Рисунок 3.2 – алгоритм проверки соответствия IPv6

# Реализация алгоритма перевода чисел в другую сс:

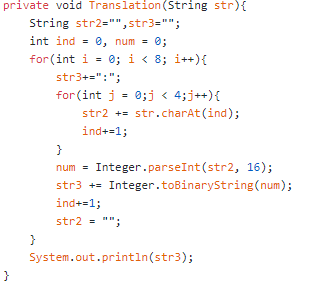


Рисунок 3.3 – перевод чисел в двоичную сс

# **Заключение**

В процессе выполнения лабораторной работы мы изучили структуру данных – строки и познакомились с классами: String, StringBuffer, Pattern, StringBuilder.

# **Список используемых источников**

1. Работа со строками[Электронный ресурс]. – URL: https://metanit.com/java/tutorial/7.1.php

# **Приложение**

Main

package com.company;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

int menu = 100;

Scanner in = new Scanner(System.in);

try {

while (menu != 0) {

System.out.print("[0] - Завершение работы" + '\n' + "[1] - Задание 1" + '\n' +

"[2] - Задание 2" + '\n' + "[3] - Задание 3" + '\n' + "Выберите задание: ");

menu = in.nextInt();

switch (menu) {

case 0:

System.out.print("Работа завершена");

break;

case 1:

task1(in);

break;

case 2:

task2(in);

break;

case 3:

task3(in);

break;

}

}

} catch(Exception ignored){

}

}

private static void task1(Scanner in){

Task1 obj = new Task1();

int menu = 100;

try {

while (menu != 0) {

System.out.print("[0] - Главное меню" + '\n' + "[1] - Ввести текст" + '\n' +

"[2] - Узнать количество абзацев, время разговора в каждом абзаце и разделить первый абзац" + '\n' +

"Выберете задание: ");

menu = in.nextInt();

switch (menu) {

case 0:

System.out.println("Возвращение в главное меню...");

break;

case 1:

System.out.println("Введите текст: ");

in.nextLine();

String str = in.nextLine();

obj.Input(str);

break;

case 2:

System.out.println("Абзацев: " + obj.CountAbz());

break;

}

}

} catch(Exception ignored){

}

}

private static void task2(Scanner in){

Task2 obj = new Task2();

int menu = 100;

try {

while (menu != 0) {

System.out.print("[0] - Главное меню" + '\n' + "[1] - Ввести текст" + '\n' +

"[2] - Удалить самый длинный абзац, представить числа в форме с плавающей точкой и выполнить модификацию текста" + '\n' +

"Выберете задание: ");

menu = in.nextInt();

switch (menu) {

case 0:

System.out.println("Возвращение в главное меню...");

break;

case 1:

System.out.println("Введите текст: ");

in.nextLine();

String str = in.nextLine();

obj.Input(str);

break;

case 2:

obj.CountAbz();

break;

}

}

} catch(Exception ignored){

}

}

private static void task3(Scanner in){

Task3 obj = new Task3();

int menu = 100;

try {

while (menu != 0) {

System.out.print("[0] - Главное меню" + '\n' + "[1] - Ввести текст" + '\n' +

"[2] - Поверить соответствие строки формату IP-адреса v4" + '\n' +

"[3] - Второе задание" + '\n' +

"Выберете задание: ");

menu = in.nextInt();

switch (menu) {

case 0:

System.out.println("Возвращение в главное меню...");

break;

case 1:

System.out.println("Введите текст: ");

in.nextLine();

String str = in.nextLine();

obj.Input(str);

break;

case 2:

if(obj.IPv4())

System.out.println("IP соответствует формату IPv4\n");

else

System.out.println("IP не соответствует формату IPv4\n");

break;

case 3:

obj.IPv6();

break;

}

}

} catch(Exception ignored){

}

}

}

Task 1

package com.company;

import java.util.ArrayList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class Task1 {

private String line = "";

private String line2 = "";

private int x = 0;

private String res = "";

private ArrayList<Double> Mass = new ArrayList<>();

void Input(String line){

if (line.charAt(line.length() - 1) != ' ')

line += ' ';

res = line;

this.line = line;

}

int CountAbz(){

int count = 0;

int temp = 0;

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

if(temp != count){

if(temp!=0) {

CountPhone(line2, temp);

Split(line2);

}

line2 = "";

temp = count;

}

else{

line2 += line.charAt(i);

}

if(line.charAt(i) == '\t')

count +=1;

}

CountPhone(line2, temp);

line = res;

line2 = "";

x = 0;

return count;

}

private void CountPhone(String line2, int temp){

String str = line2;

double sum = 0.0;

Pattern pattern = Pattern.compile("-?\\d+(\\.\\d+)?");

Matcher matcher = pattern.matcher(str);

while(matcher.find()) {

sum += Double.parseDouble(matcher.group());

}

Mass.add(sum);

System.out.println("Время разговоров в " + temp + " абзаце: " + sum);

}

private void Split(String line3) {

if(x==0){

System.out.println("Первый абзац, разбитый на предложения");

for (String predl : line3.split(". ")) {

System.out.println(predl);

}

x+=1;

}

}

}

Task 2

package com.company;

import java.util.ArrayList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class Task2 {

private StringBuilder line;

private StringBuilder line1;

private StringBuffer line2;

private StringBuffer line3;

Boolean d = false;

private ArrayList<Integer> Mass = new ArrayList<>();

private ArrayList<Integer> Mass2 = new ArrayList<>();

private String res = "";

private Pattern pattern = Pattern.compile("-?\\d+(\\.\\d+)?");

void Input(String str){

try {

if (str.charAt(str.length() - 1) != ' ')

str += " ";

line = new StringBuilder(str);

line1 = new StringBuilder(" ");

res = str;

line2 = new StringBuffer(" ");

line3 = new StringBuffer(" ");

}

catch(Exception e) {

System.out.println("Error");

}

}

void CountAbz(){

try {

int count = 0;

int temp = 0;

int temp2 = 0;

int max = 0;

String tmp = "";

for (int i = 0; i < line.length() - 2; i++) {

temp += 1;

temp2 += 1;

if (count == 1) {

tmp = String.valueOf(line.charAt(i));

line2.append(tmp);

}

if (line.charAt(i) == '\t') {

count += 1;

if (count != 1) {

Mass.add(temp);

Mass2.add(temp2);

if (temp > max)

max = temp;

}

temp = 0;

}

}

if (temp > max)

max = temp;

Mass.add(temp);

Mass2.add(temp2);

int ind = Mass.indexOf(max);

tmp = String.valueOf(line2);

if (ind != 0) {

line.delete(Mass2.get(ind - 1), Mass2.get(ind)-1);

line.delete(0, Mass2.get(0));

line.append("\t").append(tmp);

} else {

line.delete(0, Mass2.get(ind)-1);

line.append("\t").append(tmp);

}

System.out.println(line);

String str = "";

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

if (line.charAt(i) == '\t') {

str = String.valueOf(line2);

Update(str);

line2.delete(0,line2.length());

line3.delete(0,line3.length());

}

line2.append(line.charAt(i));

}

line.delete(0,line.length());

line.append(res);

line2.delete(0,line2.length());

line2.append(" ");

System.out.print(line1+"\n");

}

catch(Exception ignored) {

}

}

private void Update(String str){

double t = 0;

line3.append(str);

Matcher matcher = pattern.matcher(str);

String temp = "";

while(matcher.find()) {

t = Double.parseDouble(matcher.group());

temp = String.valueOf(t);

line3.replace(matcher.start(), matcher.start()+temp.length(), Modify(t));

}

if(d)

line1.append(line3);

d = true;

}

private String Modify(double b){

double t = Double.parseDouble(String.valueOf(b));

int a = 0;

while(true){

if(t>1){

t/=10;

a+=1;

}

else

break;

}

return t +"\*exp\*10^"+a;

}

}

Task 3

package com.company;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class Task3 {

private String line = "";

void Input(String line){

if (line.charAt(line.length() - 1) != ' ')

line += ' ';

this.line = line;

}

boolean IPv4() {

if (line == null || line.isEmpty())

return false;

line = line.trim();

if ((line.length() < 6) || (line.length() > 15))

return false;

try {

Pattern pattern = Pattern.compile("^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$");

Matcher matcher = pattern.matcher(line);

return matcher.matches();

} catch (Exception ex) {

return false;

}

}

void IPv6(){

String str="";

line = line.trim();

try {

Pattern pattern = Pattern.compile("([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}");

Matcher matcher = pattern.matcher(line);

while(matcher.find()) {

str = matcher.group();

System.out.print("IPv6: "+ str +" в двоичном представлении: ");

Translation(str);

}

}

catch (Exception ex) {

System.out.println("error");

}

}

private void Translation(String str){

String str2="",str3="";

int ind = 0, num = 0;

for(int i = 0; i < 8; i++){

str3+=":";

for(int j = 0;j < 4;j++){

str2 += str.charAt(ind);

ind+=1;

}

num = Integer.parseInt(str2, 16);

str3 += Integer.toBinaryString(num);

ind+=1;

str2 = "";

}

System.out.println(str3);

}

}