ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО образования

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

**ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ**

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**Отчет**

по дисциплине «Разработка программных приложений на языках высокого уровня»

специальности 09.03.03 «Прикладная информатика»

**Лабораторная работа № 4**

студентки очного отделения

3 курса 12002005 группы

Лысенко Натальи Александровны

Проверил:

старший преподаватель

Лифиренко Максим Вячеславович

БЕЛГОРОД 2023

**Цель работы:** получить практические навыки работы с Collections в Java, изучить работу с регулярными выражениями.

**Вариант №13**

На вход программы подается произвольный html-файл, необходимо удалить все атрибуты у всех тегов, кроме заданных пользователем.

Вывести в консоль статистику по обработанному файлу:

- указать тег и кол-во в исходном файле из заданных пользователем

- указать тег и кол-во в исходном файле у которых удалены атрибуты

Сохранить новый html-файл.

**Листинг**

import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class HtmlAttributeRemover  
{  
 */\*\*  
 \* Удаляет атрибуты у тегов HTML, кроме тех, которые указаны в excludeTags. Выводит статистику по удалению в консоль.  
 \** ***@param*** *inputPath путь к файлу с HTML-кодом, который нужно обработать  
 \** ***@param*** *outputPath путь к файлу, в который нужно записать обработанный HTML-код  
 \** ***@param*** *excludeTags список тегов, которые нужно исключить из обработки, перечисленные через вертикальную черту '|'  
 \*/* public static void removeHtmlAttributes(Path inputPath, Path outputPath, String excludeTags)  
 {  
 // Создание регулярного выражения для поиска тегов HTML с атрибутами  
 String regex = "<(?!(" + excludeTags + "))([a-z]+)(?:[^>]\*?)(?<!/)>";  
  
 // Чтение HTML-кода из файла  
 String html;  
 try  
 {  
 html = Files.*readString*(inputPath);  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Ошибка чтения файла: " + e.getMessage());  
 return;  
 }  
  
 // Удаление атрибутов у тегов  
 String cleanedHtml = Pattern.*compile*(regex).matcher(html).replaceAll(matchResult ->  
 {  
 String tagName = matchResult.group(2);  
 return "<" + tagName + ">";  
 });  
  
 // Запись HTML-кода в новый файл  
 try  
 {  
 Files.*writeString*(outputPath, cleanedHtml);  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Ошибка записи файла: " + e.getMessage());  
 return;  
 }  
  
 // Вывод в консоль списка тегов, которые не нужно обрабатывать и их количество  
 Pattern tagPattern = Pattern.*compile*("<(" + excludeTags + ")[^>]\*>");  
 Matcher tagMatcher = tagPattern.matcher(html);  
 Map<String, Integer> tagCounts = new HashMap<>();  
 while (tagMatcher.find())  
 {  
 String tagName = tagMatcher.group(1);  
 tagCounts.put(tagName, tagCounts.getOrDefault(tagName, 0) + 1);  
 }  
 if (!tagCounts.isEmpty())  
 {  
 System.*out*.println("Теги, которые не нужно обрабатывать:");  
 for (String tagName : tagCounts.keySet())  
 {  
 System.*out*.println(tagName + ": " + tagCounts.get(tagName));  
 }  
 }  
 else  
 {  
 System.*out*.println("Все теги были обработаны");  
 }  
  
 // Вывод в консоль списка тегов, у которых были удалены атрибуты и их количество  
 String regex2 = "<(?!(" + excludeTags + "|/(" + excludeTags + ")))([a-z]+)(?:[^>]\*?)(?<!/)>";  
 Pattern pattern = Pattern.*compile*(regex2);  
 Matcher matcher = pattern.matcher(html);  
 Map<String, Integer> tagCountMap = new HashMap<>();  
 while (matcher.find())  
 {  
 String tagName = matcher.group(3);  
 // Увеличиваем счетчик для данного тега  
 tagCountMap.put(tagName, tagCountMap.getOrDefault(tagName, 0) + 1);  
 }  
 System.*out*.println("Теги, у которых были удалены атрибуты:");  
 tagCountMap.forEach((tagName, count) -> System.*out*.println(tagName + ": " + count));  
  
 System.*out*.println("Файл успешно обработан");  
 }  
  
 */\*\*  
 \* Точка входа в программу.  
 \** ***@param*** *args аргументы командной строки(не используется)  
 \*/* public static void main(String[] args)  
 {  
 Path inputPath = Paths.*get*("input.html"); // Путь к входному файлу  
 Path outputPath = Paths.*get*("output.html"); // Путь к выходному файлу  
 String excludeTags = "a|header"; // Список тегов, которые не нужно обрабатывать  
 *removeHtmlAttributes*(inputPath, outputPath, excludeTags);  
 }  
}

На рисунке 1 представлен файл с входными данными.

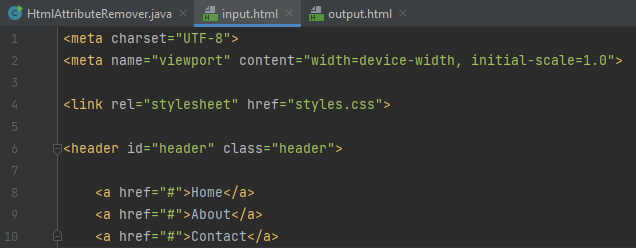


Рисунок 1 – Входной файл

На рисунке 2 представлен результат работы программы, сохраненный в файл.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Файл с результатом

На рисунке 3 представлен результат работы программы, который выводится в консоль.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Статистика по обработанному файлу

**Вывод:** были получены практические навыки работы с Collections в Java, и изучена работа с регулярными выражениями.