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 № 5**

студентки очного отделения

3 курса 12002005 группы

Лысенко Натальи Александровны

Проверил:

старший преподаватель

Лифиренко Максим Вячеславович

БЕЛГОРОД 2023

**Цель работы:** получить практические навыки создания многопоточных приложений.

**Вариант №13**

Реализовать многопоточное приложение, реализующее поиск подстроки в файлах. Список файлов передается в качестве параметра командной строки. Для каждого файла выделяется отдельный поток. Для вывода результатов поиска в консоль создается отдельный поток, считывающий данные по мере поступления из разделяемого списка объектов класса SearchResult, имеющего следующего поля «имя файла», «индекс вхождения».

**Листинг**

import java.io.\*;  
import java.util.\*;  
import java.util.concurrent.\*;  
import java.lang.System;  
public class FileSearch {  
 */\*\*  
 \* Главный метод программы.  
 \* Принимает строку для поиска в качестве аргумента командной строки.  
 \* Получает список имен файлов в текущей директории и запускает для каждого файла  
 \* отдельный поток, который ищет вхождения строки в файле.  
 \* По завершении всех потоков выводит на экран информацию о найденных вхождениях.  
 \*/* public static void main(String[] args) throws InterruptedException {  
  
 File currentDir = new File(".");  
 File[] files = currentDir.listFiles();  
 List<String> fileNames = new ArrayList<>();  
  
 // Фильтруем и добавляем только текстовые файлы  
 for (File file : files) {  
 if (file.isFile() && file.getName().endsWith(".txt")) {  
 fileNames.add(file.getName());  
 }  
 }  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.println("Введите искомую подстроку:");  
 String substring = scan.nextLine();  
  
 // Выводим список доступных файлов  
 System.*out*.println("Available files: " + fileNames);  
 // Создаем очередь для хранения результатов поиска и список потоков  
 BlockingQueue<SearchResult> results = new LinkedBlockingQueue<>();  
 List<Thread> threads = new ArrayList<>();  
  
 // Создаем поток для каждого файла и запускаем его  
 for (String name : fileNames) {  
 Thread thread = new Thread(new FileSearcher(substring, name, results));  
 thread.start();  
 threads.add(thread);  
 }  
  
 // Поток для чтения списка с результатами и вывода их на экран  
 Thread outputThread = new Thread(() -> {  
 try {  
 while (!Thread.*interrupted*())  
 {  
 SearchResult result = results.take();  
 System.*out*.printf("Found '%s' in file '%s' at line %d, position %d\n",  
 result.substring, result.fileName, result.lineNumber, result.index);  
 }  
 } catch (InterruptedException ignored) {}  
 });  
 outputThread.start();  
 Thread.*sleep*(1000);  
 outputThread.interrupt();  
 // Ожидаем завершения всех потоков  
 for (Thread thread : threads) {  
 thread.join();  
 }  
 }  
  
 */\*\*  
 \* Класс для поиска вхождений строки в файле в отдельном потоке.  
 \*/* static class FileSearcher implements Runnable {  
 private final String fileName;  
 private final String substring;  
 private final BlockingQueue<SearchResult> results;  
 */\*\*  
 \* Конструктор класса.  
 \*  
 \** ***@param*** *substring строка, которую нужно найти.  
 \** ***@param*** *fileName имя файла, в котором нужно искать.  
 \** ***@param*** *results очередь для хранения результатов поиска.  
 \*/* public FileSearcher(String substring, String fileName, BlockingQueue<SearchResult> results) {  
 this.substring = substring;  
 this.fileName = fileName;  
 this.results = results;  
 }  
 @Override  
 public void run() {  
 try {  
 // Создаем объект BufferedReader для чтения файла построчно  
 BufferedReader reader = new BufferedReader(new FileReader(fileName));  
 String line;  
 int lineNumber = 1;  
 // Ищем подстроку в каждой строке файла  
 while ((line = reader.readLine()) != null) {  
 int index = line.indexOf(substring);  
 if (index >= 0) {  
 // Если найдено совпадение, добавляем результат в очередь  
 results.put(new SearchResult(substring, fileName, index, lineNumber));  
 }  
 lineNumber++;  
 }  
 reader.close();  
 } catch (IOException | InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 */\*\*  
 \* Класс SearchResult представляет собой результат поиска вхождения строки в файле.  
 \* Он содержит информацию о найденной подстроке, имени файла, индексе первого вхождения  
 \* и номере строки, в которой найдено вхождение.  
 \*/* private static class SearchResult {  
  
 private final String substring;  
 private final String fileName;  
  
 private final int index;  
 private final int lineNumber;  
  
 */\*\*  
 \*Создает новый объект SearchResult с заданными параметрами.  
 \** ***@param*** *substring найденная подстрока.  
 \** ***@param*** *fileName имя файла, в котором найдено вхождение.  
 \** ***@param*** *index индекс первого вхождения подстроки в строке файла.  
 \** ***@param*** *lineNumber номер строки файла, в которой найдено вхождение.  
 \*/* public SearchResult( String substring, String fileName, int index, int lineNumber) {  
 this.substring = substring;  
 this.fileName = fileName;  
 this.index = index;  
 this.lineNumber = lineNumber;  
 }  
 }  
}

На рисунке 1 представлены файлы с исходными данными

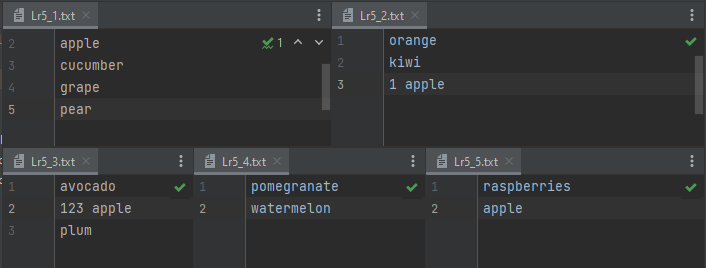


Рисунок 1 – Файлы с данными для поиска

Передаем в качестве параметра имена файлов, в которых будем искать подстроку. (Рисунок 2)

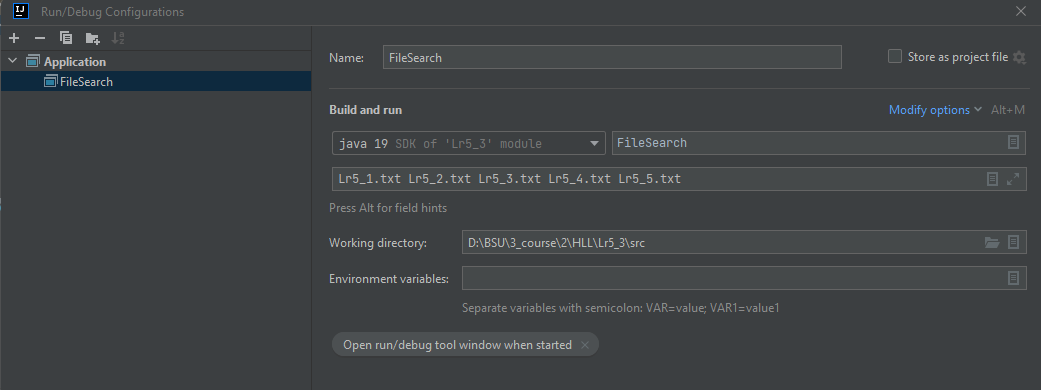


Рисунок 2 – Передаваемые параметры

На рисунке 3 представлен результат работы программы.

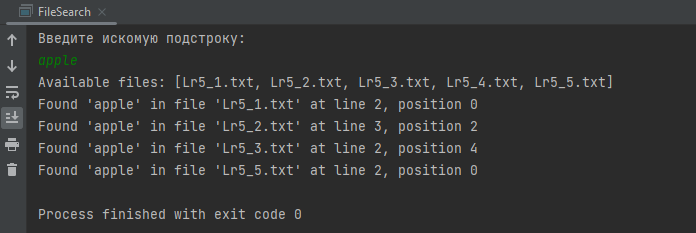


Рисунок 3 – Результат в консоли

**Вывод:** получили практические навыки создания многопоточных приложений.