Programowanie w języku JAVA

Laboratorium 5

Marcin Godfryd grupa 31

import java.io.\*;  
import java.nio.file.\*;  
import java.util.\*;  
import java.util.concurrent.\*;  
import java.util.regex.\*;  
  
public class WordCount {  
  
 public static void main(String[] args) {  
 if (args.length != 2) {  
 System.*err*.println("java WordCount <number\_of\_threads> <file\_path>");  
 return;  
 }  
  
 int numberOfThreads;  
 try {  
 numberOfThreads = Integer.*parseInt*(args[0]);  
 } catch (NumberFormatException e) {  
 System.*err*.println("Liczba wątków musi być liczbą całkowitą.");  
 return;  
 }  
  
 String filePath = args[1];  
 Path file = Paths.*get*(filePath);  
 if (!Files.*exists*(file)) {  
 System.*err*.println("Taki plik nie istnieje: " + filePath);  
 return;  
 }  
  
 try {  
 new WordCounter(numberOfThreads, file).countWords();  
 } catch (IOException | InterruptedException e) {  
 System.*err*.println("Błąd działania aplikacji: " + e.getMessage());  
 }  
 }  
}  
  
class WordCounter {  
 private final int numberOfThreads;  
 private final Path file;  
 private final ConcurrentHashMap<String, Integer> wordCounts = new ConcurrentHashMap<>();  
 private final ExecutorService executor;  
 private final ConcurrentHashMap<Integer, Integer> linesPerThread = new ConcurrentHashMap<>();  
  
 public WordCounter(int numberOfThreads, Path file) {  
 this.numberOfThreads = numberOfThreads;  
 this.file = file;  
 this.executor = Executors.*newFixedThreadPool*(numberOfThreads);  
 }  
  
 public void countWords() throws IOException, InterruptedException {  
 List<String> lines = Files.*readAllLines*(file);  
  
 for (int i = 0; i < lines.size(); i++) {  
 int threadIndex = i % numberOfThreads;  
 executor.execute(new WordCountTask(lines.get(i), threadIndex, wordCounts, linesPerThread));  
 }  
  
 executor.shutdown();  
 executor.awaitTermination(1, TimeUnit.*HOURS*);  
  
 printResults();  
 }  
  
 private void printResults() {  
 linesPerThread.forEach((threadId, lineCount) -> System.*out*.println("Thread " + threadId + ": " + lineCount));  
 wordCounts.entrySet().stream()  
 .sorted(Map.Entry.*comparingByKey*())  
 .forEach(entry -> System.*out*.println(entry.getKey() + " " + entry.getValue()));  
 }  
}  
  
class WordCountTask implements Runnable {  
 private final String line;  
 private final int threadIndex;  
 private final ConcurrentHashMap<String, Integer> wordCounts;  
 private final Map<Integer, Integer> linesPerThread;  
  
 public WordCountTask(String line, int threadIndex, ConcurrentHashMap<String, Integer> wordCounts, Map<Integer, Integer> linesPerThread) {  
 this.line = line;  
 this.threadIndex = threadIndex;  
 this.wordCounts = wordCounts;  
 this.linesPerThread = linesPerThread;  
 }  
  
 @Override  
 public void run() {  
 String[] words = line.split("\\s+");  
 for (String word : words) {  
 word = word.replaceAll("[^a-zA-Z]", "").toLowerCase();  
 if (word.length() > 1) {  
 wordCounts.merge(word, 1, Integer::*sum*);  
 }  
 }  
  
 linesPerThread.merge(threadIndex, 1, Integer::*sum*);  
 }  
}



