МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №1

з дисципліни «Сучасні технології веб-застосувань на платформі Java»

на тему *«Дослідження багатопоточності в Java з пакетом CONCURRENCY»*

Виконав:
студент групи ІО-06
Лі Юемін
Номер залікової книжки: №0672

Перевірила:

Київ 2022

Дисципліна "Сучасні технології веб-застосувань на платформі Java"

Лабораторна робота № 1

ДОСЛІДЖЕННЯ БАГАТОПОТОЧНОСТІ В JAVA З ПАКЕТОМ CONCURRENCY

Мета роботи: Ознайомитися з організацією багатопоточності в мові Java, дослідити створення потоку через розширення інтерфейсів **Runnable** і **Callable** та використання пулів потоку (*executors*).

Завдання: Написати програму на мові Java, яка рекурсивно обробляє вказану/введену з клавіатури директорію згідно варіанту (таблиця), запускаючи для обробки кожної її піддиректорії окремий потік та використовуючи для цього можливості *executors*. Файли директорії/піддиректорії також оброблюються у окремих потоках.

Методичні вказівки:

Для забезпечення можливості проведення пошуку файлів необхідно назву директорії, в якій здійснюється пошук файлів, назву нових директорії та/або файлу для запису результату пошуку, якщо це задано варіантом, вводити набором на клавіатурі.

Для забезпечення стійкості роботи програми передбачити перевірку коректності введення даних та запропонувати вирішення у разі невірного їх введення для продовження роботи програми.

Всі дії програми фіксувати відповідними повідомленнями.

11	Знайти в заданій директорії текстові файли і підрахувати в кожному файлі кількість слів, що починаються із заданої літери. В деякий файл записати назви таких файлів і кількість знайдених слів. Вивести в консоль вміст створеного файлу

Виконання роботи

Для зручності я створив на робочому столі папку, назвав її teset. За допомогою моєї програми я створив файли для перевірки та кінцевий файл для результатів.

Код програми

Main.java

```
package JW_LAB1;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws IOException{

        try {
            FileUtils.generateFiles();
        } catch (IOException ex) {
            System.out.printf("Cannot generate files: %s%n",
ex.getMessage());
        }

        System.out.printf("%nLab1 started%n%n");

        new Runner().run();

        System.out.printf("%nLab1 finished%n");


    }



}
```

Runner.java

```
package JW_LAB1;

import java.io.File;
import java.util.Scanner;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class Runner {
```

```
    public void run() {
        Scanner scanner = new Scanner(System.in);
        File file;

        do {
            System.out.print("Enter a directory path: ");
            file = new File(scanner.nextLine());
            System.out.println();
        } while (!correctInputDir(file));

        System.out.print("Enter a main letter to match with: ");
        String mainLetter = new String(scanner.nextLine());
        File fileResult = new
File("C:\\Users\\artisoul0\\Desktop\\teset\\resultfile.txt");
        long start = System.currentTimeMillis();

        ExecutorService pool = Executors.newCachedThreadPool();
        Task task = new Task(file, pool, mainLetter, fileResult);
        Future<TaskResult> result = pool.submit(task);

        try {
            result.get();
        } catch (ExecutionException | InterruptedException ex) {
            ex.printStackTrace();
        }

        System.out.printf("Total time: %d ms%n", System.currentTimeMillis() -
start);

        pool.shutdown();
    }

    private boolean correctInputDir(File file) {
        if (!file.exists() || !file.isDirectory()) {
            System.out.println("Provided directory does not exist. Try
again");
            return false;
        }
        return true;
    }
}
```

FileUtils.java

```
package JW_LAB1;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;

public class FileUtils {
    public static void generateFiles() throws IOException{
        File file1 = new
File("C:\\Users\\artisoul0\\Desktop\\teset\\file1.txt");
        forceCreate(file1);
        writeText(file1, """
                The United States media landscape has been characterised by
```

```
the centrality of large-scale cultural industries since the development of
the penny press in the 1830s. For a while in the nineteenth century big urban
newspapers were the largest manufacturing companies in the country. This
trend continued with the rise of Hollywood, commercial broadcasting and
associated industries like music recording and advertising. \s""");


        File file2 = new
File("C:\\Users\\artisoul0\\Desktop\\teset\\file2.txt");
        forceCreate(file2);
        writeText(file2, """
                For a number of decades in the mid-twentieth century a fairly
stable equilibrium existed in the media system, with strong stable markets
that made the dominant media companies highly profitable and very influential
as social institutions. Newspapers, broadcast companies and magazines all
invested heavily in newsrooms and the profession of journalism grew in
number, autonomy and influence. Journalism was characterised by a low level
of "political parallelism," with the "objectivity norm" dominating
journalistic ethics and most news organisations avoiding identification with
particular political parties or tendencies.
                """);


        File resultFile = new
File("C:\\Users\\artisoul0\\Desktop\\teset\\resultfile.txt");
        forceCreate(resultFile);
        writeText(resultFile, """

                """);

    }
    private static void forceCreate(File file) throws IOException {
        if (file.exists()) {
            Files.delete(file.toPath());
        }
        Files.createFile(file.toPath());
    }

    private static void writeText(File file, String text) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(file))) {
            writer.write(text);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public static int findAndMarkMatch(File file, String mainLetter) {
        int count = 0;
        int deletedWords = 0;
        try (
                BufferedReader reader = new BufferedReader(new
FileReader(file));
                BufferedWriter writer = new BufferedWriter(new
FileWriter(file, true))
        ) {
            StringBuilder result = new StringBuilder();
            String[] words = null;
            String line;
            while ((line = reader.readLine()) != null) {

                words = line.split(" ");
                for (int j = 0; j < words.length; j++) {
                    if
```

```
(mainLetter.equalsIgnoreCase(getFirstLetterWord(words[j]))) {
                        count++;
                    }
                }

                result.append(count);
            }

            writer.write(result.toString());
        } catch (IOException ex) {
            throw new RuntimeException("Error while matching words from %s :
%s".formatted(file.getAbsolutePath(), ex.getMessage()));
        }
        return count;
    }

    public static String getFirstLetterWord(String word) {
        String[] letters = word.split("");
        return letters[0];
    }
}
```

Task.java

```
package JW_LAB1;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Future;

public class Task implements Callable<TaskResult> {
    private final File file;
    private final ExecutorService pool;

    private final String mainLetter;
    private final File fileResult;

    public Task(File file, ExecutorService pool, String mainLetter, File
fileResult) {
        this.file = file;
        this.pool = pool;
        this.mainLetter = mainLetter;
        this.fileResult = fileResult;
    }

    @Override
    public TaskResult call() {

        if (!file.isDirectory()) {
            int matchedWordsCount = FileUtils.findAndMarkMatch(file,
```

```
mainLetter);
            System.out.printf("File %s : matched %d words%n",
file.getAbsolutePath(), matchedWordsCount);
            writeText(fileResult, "File " + file.getAbsolutePath() + "
matched words: " +  matchedWordsCount);
            return new TaskResult(matchedWordsCount);
        }

        List<Future<TaskResult>> dirResults = new ArrayList<>();

        for (File file : file.listFiles()) {
            if (file.isDirectory() || (!file.isDirectory() &&
isTxtFile(file)) && !file.getName().equals("resultfile.txt")) {
                Future<TaskResult> result = pool.submit(new Task(file, pool,
mainLetter, fileResult));
                dirResults.add(result);
            }
        }

        TaskResult result = collectResults(dirResults);
        System.out.printf("Directory %s : matched %d words in total%n",
file.getAbsolutePath(), result.getMatchedWords());
        writeText(fileResult, "Directory " + file.getAbsolutePath() + "
matched words in total: " +  result.getMatchedWords());
        System.out.println(result.getMatchedWords());
        return result;
    }

    private boolean isTxtFile(File file) {
        String[] fileParts = file.toPath().toString().split("\\.");
        return fileParts[fileParts.length - 1].equals("txt");
    }

    private static void writeText(File file, String text) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(file,true))) {
            writer.write(text + '\n');
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    private TaskResult collectResults(List<Future<TaskResult>> dirResults) {
        TaskResult collected = new TaskResult();
        dirResults.forEach(labResultFuture -> {
            try {

collected.addMatchedWords(labResultFuture.get().getMatchedWords());
            } catch (InterruptedException | ExecutionException ex) {
                System.out.printf("Error while collecting results: %s%n",
ex.getMessage());
                throw new RuntimeException(ex);
            }
        });
        return collected;
    }
}
```

TaskResult.java

```java
package JW_LAB1;

public class TaskResult {
    private Integer countwords;

    public TaskResult() {
        countwords = 0;
    }

    public TaskResult(Integer countwords) {
        this.countwords = countwords;
    }

    public void addMatchedWords(int count) {
        countwords += count;
    }

    public Integer getMatchedWords() {
        return countwords;
    }
}
```

Результати виконання у консолі

Lab1 started

Enter a directory path: C:\Users\artisoul0\Desktop\teset

Enter a main letter to match with: H

File C:\Users\artisoul0\Desktop\teset\file1.txt : matched 2 words

File C:\Users\artisoul0\Desktop\teset\file2.txt : matched 2 words

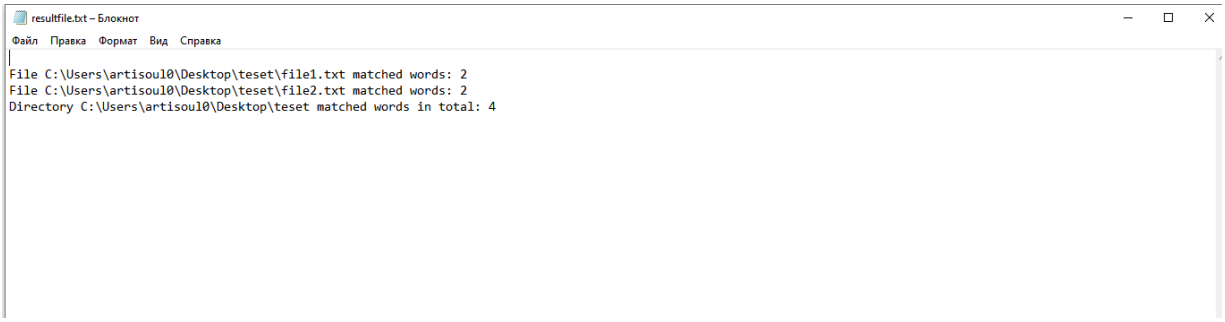
Directory C:\Users\artisoul0\Desktop\teset : matched 4 words in total 4

Total time: 28 ms

Lab1 finished

Process finished with exit code 0

Результати виконання у документі



Висновок: під час виконання лабораторної роботи №1, я вивчив, як правильно створювати багатопоточну програму в мові Java та ознайомився зі створенням потоків через розширення інтерфейсів Runnable та Callable та пулів потоку.