

# Assignment Cover Sheet / Declaration of Originality

Last name:	Dissanayake	Student ID:	20546487
Other name(s):	Dissanayake Mudiyanseelage Osura Ronath Dissanayake		
Unit name:	Software Engineering Concepts	Unit ID:	COMP3003
Lecturer / unit coordinator:	Dr David Cooper	Tutor:	Dr Shyam Reyal
Date of submission:	19/9/21	Which assignment?	SEC Assignment 1

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: Osura Ronath

Date of signature: 19/9/21

# Table of Contents

Introduction .....	3
Design .....	3
__ ComparisonResult.java.....	3
__ FileHandling.java .....	3
__ CalcSimilarity.java.....	4
__ MainClass.java.....	4
Scalability .....	5
Non-Functional Requirements.....	5
Solution .....	5
Referencing and Permissions.....	6
References.....	7

## Introduction

The readme file contains the instructions which are required to setup the project. Also, each significant piece of code has been accompanied by comments that explain the purpose and use of the code. In this document, I will address how the threads are implemented and how they contribute to the overall project improvement.

## Design

The JavaFX program contains 4 classes.

- MainClass.java
- FileHandling.java
- CalcSimilarity.java
- ComparisonResult.java

### ComparisonResult.java

The default class given to us. This class contains the constructors, getters and setters required to get the filenames and the corresponding similarity scores to the MainClass class for execution.

### FileHandling.java

This class can be considered the engine of the whole project. It contains 5 methods that will help find the required non-empty files and compare them with each other to return them to the main class to be displayed on the table. The 2 main methods which contains the multithreading concepts are described below.

#### **public synchronized void threadFileHandler**

This runnable method was used to find and filter all of the file in a given directory which are non-empty and have the required file extensions. LinkedBlockingQueue was set up in order to do the necessary step of putting all filtered files into the queue, ultimately its used for thread communication. This can be achieved by using the code queue.put. I have also initiated a POISON file to determine the end of the file, this was possible by utilizing finally within try catch.

#### **public synchronized void threadCompareHandler**

Utilizing executorService, this method creates thread pools, which execute the runnable class. This has been achieved using the newCachedThreadPool as we start with 0 threads and grow as the system resources and potential numbers of files being compared allow up until there is integer.MAX\_VALUE. We used thread pools to create reusable threads instead of creating a new one every time we compared files, as it is a means of handling threads that is far more efficient. I have chosen

newCachedThreadPool as it will not reach a saturation point of thread creation even when the load is exceptionally high. The system will also throw an InterruptedException if the available resources are insufficient for the task. This will interrupt the ongoing thread and cause the threads to stop.

I have utilized a for loop to execute the file comparisons by taking the files from the queue which we contain all of the filtered files as discussed in the previous method. This was achieved by using queue.take. If 2 or more files have been detected, then those files are passed on to the loop where the latest file which was received is compared by the all the previous files that are present until it detects the POISON file in which the loop breaks. This method will ensure that duplicate comparisons will not occur.

### **public void threadStop**

I have used this method to call the interrupts for both the threads which are being used in this project. Therefore, we can call this method in the main method where the stopComparison method is present. Upon clicking the stop button in the UI, this will interrupt all the thread activity which is ongoing.

Both runnable methods are synchronized to provide concurrency and minimize race conditions and prevent deadlocks [2].

## **CalcSimilarity.java**

This class contains the algorithm required to perform the file comparison task. After the files are filtered in the FileHandling class, they are converted into a char array and passed on to CalcSimilarity.

## **MainClass.java**

This is another default class provided to us by which contains the UI elements. In this class, the necessary constructors are called which references the data that's being passed from the FileHandling class to the MainClass to be displayed on the UI by using Platform.runLater. However only the comparison which have a similarity score of greater than 0.5 will be displayed as the rest of the comparisons are stored on the result.csv file. Furthermore, the progress bar works in synchronization with the file uploads.

## Scalability

Scalability problems can occur when the number of files being processed significantly increases. Having to process more files will require more system resources to ensure the program runs correctly. This program is thread-based and heavily relies on thread pools. Thus, if more files need to be compared, then a greater number of threads will be needed. According to the device and its specifications, if there are more threads than the processor can handle, the processor will encounter difficulties handling the job and will likely throw an exception. The program will be terminated.

## Non-Functional Requirements

I have explained in the preceding paragraph why if we don't meet the scalability requirement, it will result in exceptions being thrown and potentially a system crash.

Consequently, performance issues will be **occurred** due to the inability to obtain adequate resources. Due to the performance issues that may arise when there are more files, more context switching can result in a user experiencing a longer processing time and slower response times.

The system will become unreliable if it is unable to keep up with the demand on a regular basis and continues to crash and experience slower speeds. The computer program may be deemed unreliable if it is not operating optimally or is unusable.

## Solution

Even though I've provided options to filter data and synchronize methods, we could further improve the code by implementing a less complex / lite algorithm [3] which would require fewer resources. Assuming that we can find the appropriate algorithm, this will be an assumption. My specific code converts the files into a char array which is sent to the CalcSimilarity class to be compared and sent to the UI to be displayed and the rest to be printed on to a CSV. Having an algorithm which is lighter and does not require a conversion to char array would allow us to save system resources.

Alternatively, we can have a user / file limiter employed into the program. Utilizing a file limiter will greatly reduce issues with thread creation and resource management, as the file limit contains the most files that can be input safely into the program. In online server platforms and most websites (such as keelssuper.com), this sort of technique is used to mitigate web traffic and improve the user experience.

# Referencing and Permissions

**From:** David Cooper <david.cooper@curtin.edu.au>  
**Sent:** 17 September 2021 13:44  
**To:** Osura Ronath Dissanayake Mudiyansele <o.dissanay@student.curtin.edu.au>  
**Cc:** shyam.r@my.sliit.lk <shyam.r@my.sliit.lk>  
**Subject:** Re: Regarding SEC Assignment 1

Hi Osura,

I'm happy if you want to use OpenCSV. :)

- Dave

On 17/9/21 3:52 pm, Osura Ronath Dissanayake Mudiyansele wrote:

Hi Dr Cooper,

I hope you're doing well during these troubling times.

I am a Sri Lankan student who is doing the Software Engineering concepts module and I am emailing you regarding the upcoming assignment 1.

Firstly, I would like to thank you for considering and accepting an extension for this assignment. We were informed to get your permission in the case we were using any third-party libraries to aid our project and provide a proof of permission in our reports.

I have used **OpenCSV** to aid me with the CSV generation of my project. I understand PrintWriter method (as implemented on WS2) can be used as well but I received an error which prevented me from implementing it fully in my project. As I was searching for alternative ways to combat the issue, I discovered this method of printing values into a CSV using OpenCSV. I have thereby implemented this on my project, and it is working as expected. I am seeking permission to use it in my project, I hope this won't be an issue to use.  
Link is referenced below.

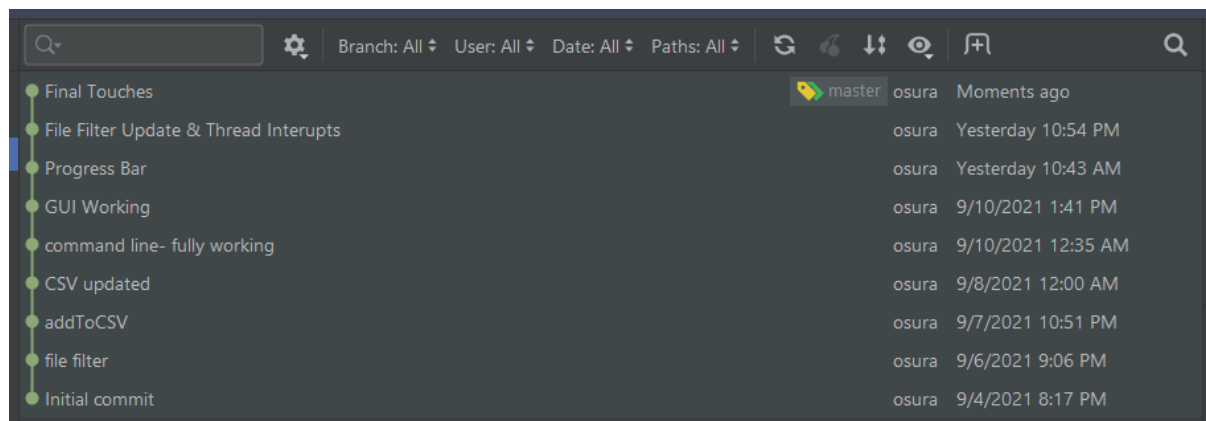
<https://www.baeldung.com/opencsv>

Kind Regards  
Osura Ronath  
20546487

Dr David Cooper  
Lecturer | Discipline of Computing, School of EECMS

Curtin University  
Email | [David.Cooper@curtin.edu.au](mailto:David.Cooper@curtin.edu.au)  
Web | [curtin.edu.au](http://curtin.edu.au)

## Proof of permission for the usage of CSV Writer [1]



	Branch: All	User: All	Date: All	Paths: All			
Final Touches		osura	Moments ago				
File Filter Update & Thread Interrupts		osura	Yesterday 10:54 PM				
Progress Bar		osura	Yesterday 10:43 AM				
GUI Working		osura	9/10/2021 1:41 PM				
command line- fully working		osura	9/10/2021 12:35 AM				
CSV updated		osura	9/8/2021 12:00 AM				
addToCSV		osura	9/7/2021 10:51 PM				
file filter		osura	9/6/2021 9:06 PM				
Initial commit		osura	9/4/2021 8:17 PM				

## Proof of work – Commit history

## References

[1] Synchronised methods - Tutorialspoint

<https://www.tutorialspoint.com/can-we-synchronize-a-run-method-in-java#:~:text=Yes%2C%20we%20can%20synchronize%20a,by%20a%20single%20thread%20only.&text=It%20is%20good%20practice%20to,threads%20at%20the%20same%20time>

[2] LCS Algorithm – Geeks for Geeks

<https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/>

[3] OpenCSV - Baeldung

<https://www.baeldung.com/opencsv>

[4] Linked Blocking Queue - GeeksforGeeks

<https://www.geeksforgeeks.org/linkedbackingqueue-class-in-java/>