AP Report

# Status Report

Successful compilation using javac, java se 7 and above. It accepts the supplied input as specified in the assignment brief. It accepts a pattern followed by zero or more directories. If no directories are specified, then it takes the root folder where the solution file – fileCrawler.java – is in to search.

The implementation of the solution uses multiple workers in a producer/consumer system to traverse through the specified directory and find any files that matches the supplied pattern.

Testing the solution and the speeds of the producer and the consumers, the solution runs faster as more consumers threads are created but starts to degrade as the number of threads start to be more than the system can handle. Results will be posted later in the report.

Thread safe structures have been used.

# Bugs and Limitation

As no specific Java version has been mentioned in the assignment brief, Java SE 7 or higher has been used in my implementation of the solution. The linux VM from UoG used for testing also only has Java SE 7 installed.

By measuring the completion timings of the producer, consumer and main thread, it was found out that the usage of the File class from Java library JAVA.IO.FILE has a significant impact on read speeds and is constantly causing the producer to be a bottleneck. This leaves the consumers idle quite often while the producer spends some time processing the File class.

Instead I have switch to using the relatively new JAVA.NIO class mainly the DirectoryStream and Files classes. Using these classes provides a significant increase in producer and consumer performance. Thus, allowing the producers to work faster and multiple consumers to make an actual difference.

However, it should be noted and emphasized that JAVA.NIO class availability is on Java SE 7 and higher.

# Results

# Rubrics

## Workable Solution

Implemented Multiple Workers

## Correct Argument Processing

The pattern is correctly read.

## Correct Processing of CRAWLER\_THREADS

Using export CRAWLER\_THREADS=<number>, the number of consumer threads can be set for the solution to declare.

## For Successful Compilation

Program compiles with JAVAC

## For Successful Compilation With No Warnings

## Reasonable, Concurrency-Safe Class Used For Work Queue

## Reasonable, Concurrency-Safe Class Used For Another Queue

## Effective Mechanism For Determination When No More Directories

## Efficient Mechanism For Determination When Worker Thread Has Finished

## If It works Correctly With The Files In The Test Folder

## If It Works Correctly With The Files In An Unseen Folder Of Files

## Runtime Performance With 1 Worker On Test Folder Is Similar To Single Threaded Implementation

## Runtime Performance On Test Folder First Improves, Then Degrades As Number Of Threads Is Increased