CMSC 335
Object-Oriented and Concurrent Programming:

Project 4

Instructor: *Professor Michael Pilone*
Student: *Dakin T. Werneburg*

July 10, 2016

TABLE OF CONTENTS

Question Analysis

Purpose of Assignment

The assignment was to build off from project 3 by using an ExecutorService that does all the thread management by implementing an ExecutorDataProcessor and to implement a new DataRetriever interface but to get the data files from an internet source. It should retrieve a Future should not freeze the GUI

Questions to be Answered

- How to get files from an HTTP source?
- How ExectorServcies work?
- How to achieve thread safety?
- How to keep track of the order of data files processed.

Program Design

Design Decisions

- The GUI is designed using JavaFX and inner classes are done using Lambda expressions.
- When implementing the DataRetriever interface I decided to use the suggested InputStream and FileOutputStream classes with the suggested buffer size of 1024.
- Generated random number using the ThreadLocalRandom class since its thread safe and has optimal performance.
- Used the executorservie shutdown method to shut down the worker threads because the threads should finish executing the threads before shutdown.
- Updated the logTextArea to occupy the CENTER of the borderlayout. This allowed the logTextArea to grow as the size of the window changes.
- Instead of removing elements from the array in the MainWindow class, replaced it with set to null, this way I could keep track of the order that items was processed.

Assumptions

- The application and GUI should do the same thing except that files are from the Internet and the threads are managed by the ExectorService.

Screen Shots

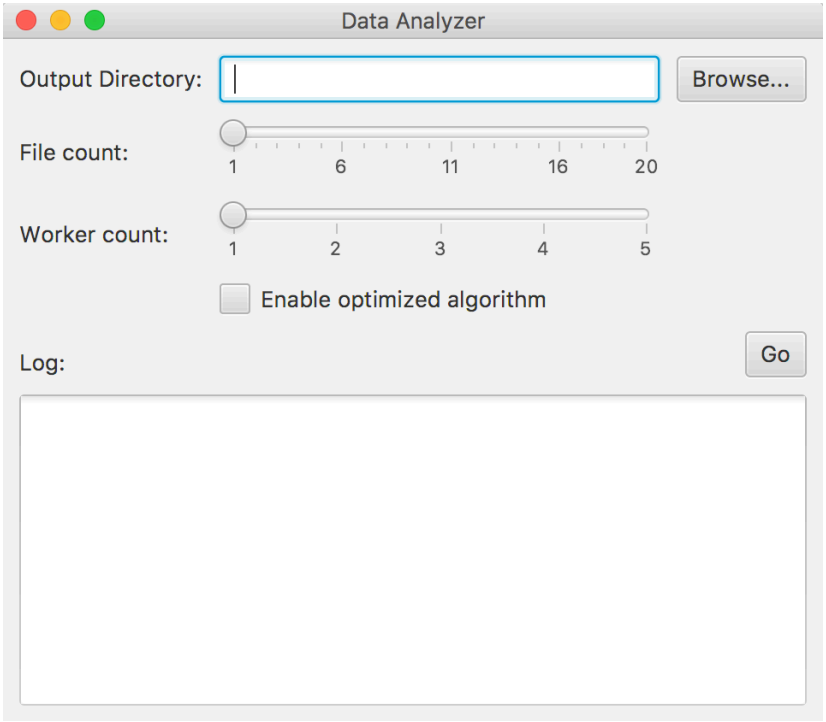


Figure 1: Initial Window

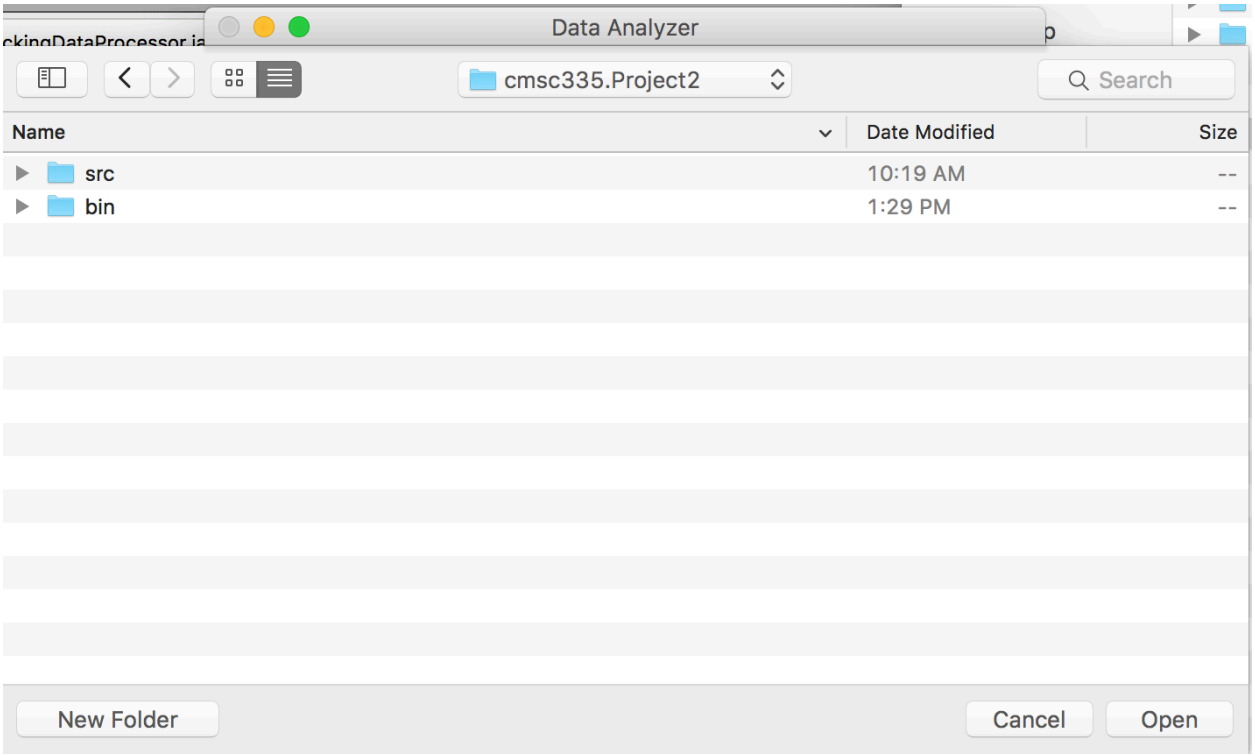


Figure 2: After Selecting "Browse.." Button

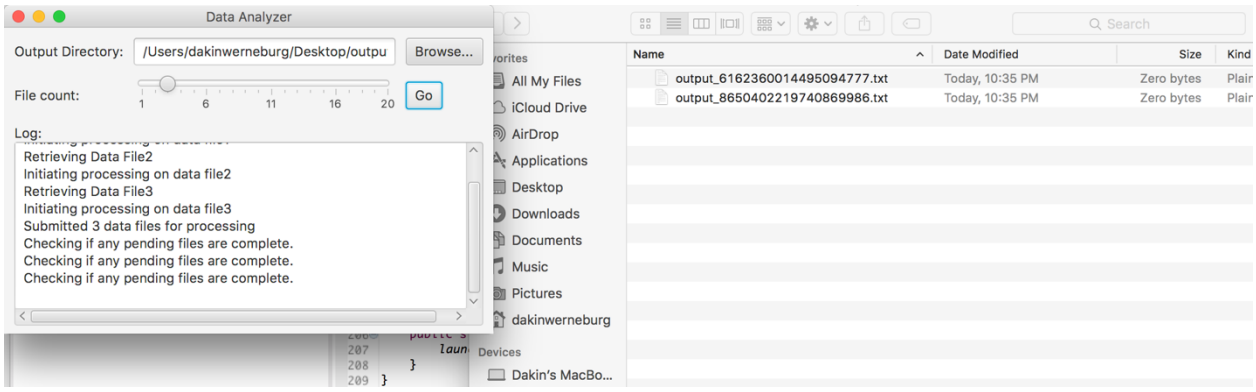


Figure 3: After Selecting Directory and go

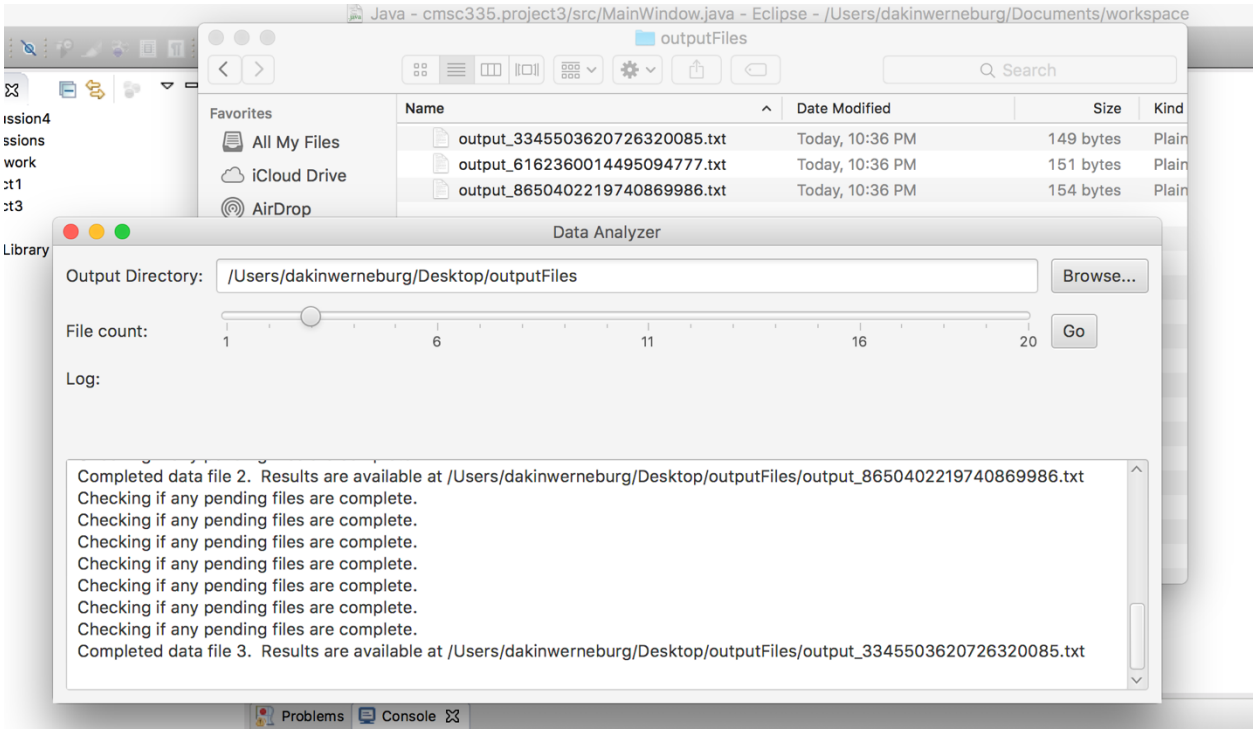
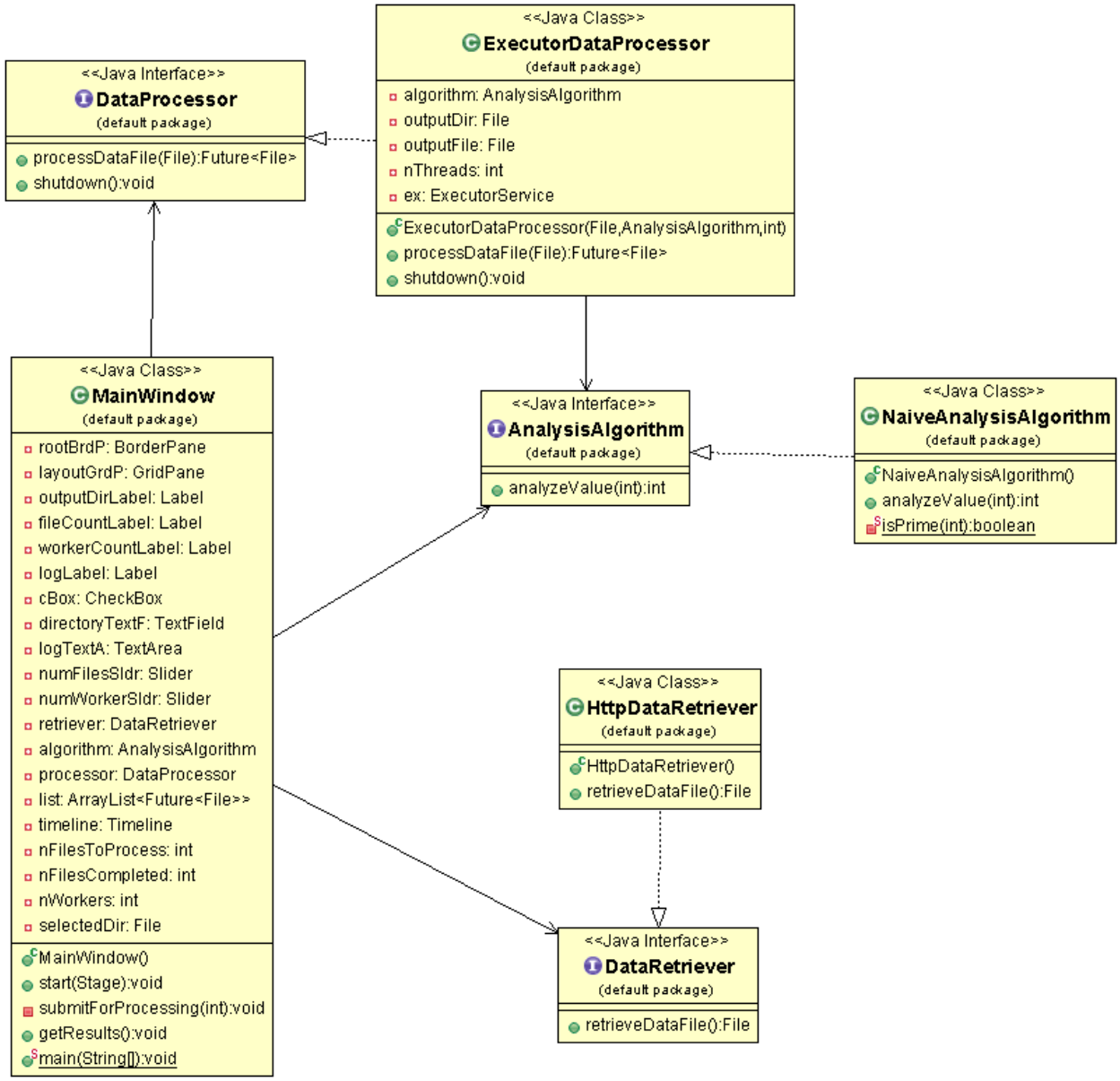


Figure 4: After files were complete

Possible Improvements

- Could have handle error handling with pop up widows vs printstacktrace
- Broken down the code into more tasks creating fewer lines of code for each method.
- Change the default styles and implemented different colors and fonts.
- Shutdown the program more cleanly. Once files are done, a window to ask if you want to continue

UML Diagram


<<Java Class>>
ExecutorDataProcessor
(default package)
algorithm: AnalysisAlgorithm
outputDir: File
outputFile: File
nThreads: int
ex: ExecutorService
ExecutorDataProcessor(File,AnalysisAlgorithm,int)
processDataFile(File):Future<File>
shutdown():void
<<Java Interface>>
DataProcessor
(default package)
processDataFile(File):Future<File>
shutdown():void
<<Java Class>>
MainWindow
(default package)
rootBrdP: BorderPane
layoutGrdP: GridPane
outputDirLabel: Label
fileCountLabel: Label
workerCountLabel: Label
logLabel: Label
cBox: CheckBox
directoryTextF: TextField
logTextA: TextArea
numFilesSldr: Slider
numWorkerSldr: Slider
retriever: DataRetriever
algorithm: AnalysisAlgorithm
processor: DataProcessor
list: ArrayList<Future<File>>
timeline: Timeline
nFilesToProcess: int
nFilesCompleted: int
nWorkers: int
selectedDir: File
MainWindow()
start(Stage):void
submitForProcessing(int):void
getResults():void
main(String[]):void
<<Java Interface>>
AnalysisAlgorithm
(default package)
analyzeValue(int):int
<<Java Class>>
NaiveAnalysisAlgorithm
(default package)
NaiveAnalysisAlgorithm()
analyzeValue(int):int
isPrime(int):boolean
<<Java Class>>
HttpDataRetriever
(default package)
HttpDataRetriever()
retrieveDataFile():File
<<Java Interface>>
DataRetriever
(default package)
retrieveDataFile():File

Test Plan

Input	Expected Output	Actual Output	Passed	Corrections made
File count set to 5 and worker count set to 3 Optimizer not checked	5 files retrieved and processed on 3 threads. Results displayed as they are ready	File not found exception	N	Forgot to add directory. Created Alert dialog once go was submitted
File count set to 5 and worker count set to 3 Optimizer not checked	5 files retrieved and processed on 3 threads. Results displayed as they are ready	As expected	Y	
File count set to 11 and worker count set to 1 Optimizer checked	11 files retrieved and processed on 1 threads. Results displayed as they are ready	As expected	Y	
File count set to 20 and worker count set to 3	20 files retrieved and processed on 3 threads. Results displayed as they are ready	As expected	Y	
Type "/Users/dakinwerneburg	Files created at that directory	As expected	Y	
Click "Browse.." button	GUI popup that get only the directory from the user not files	As expected	Y	

Error Handling

- **Exception** – printStackTrace()
- **IOException** - printStackTrace()
- **InterruptedException -** printStackTrace()
- **ExecutionException** - printStackTrace()
- **FileNotFoundException -** printStackTrace() and Alert dialog class used to tell user that they need to enter a directory at the main window when they select go.

Lessons Learned

- ExecutorService makes multithreading much safer and easier to manage, especially for large number of threads.
- InputStream reads one bit at a time and the FileOutputStream send the input to a File. When using a buffer to read data, you must flush it just in case the download is incomplete or error is reading the file. All sources need to be closed.
- URL is a pointer to source on the web
- This project was actually easier since the working of thread management was done by the ExecutorService.
- If you run a thread in the event dispatch thread (EDT), you will block the UI. So all object instantiations or methods that call run but be outside the EDT
- Lambda expression are awesome and make coding simplified. It requires just the parameter and the expression and based on how many parameters and expression less things are needs such as parentheses and curly braces. This make reading the code much easier to follow [1].

References

1. *Java 8 Lambda Expression.* Retrieved from http://programmers.stackexchange.com/questions/141563/should-main-method-be-only-consists-of-object-creations-and-method-calls