Please replace this paragraph with your Assignments 1-4 unchanged but including grading and comments, even if this assignment is separate. Please note the addition of the evaluation criterion “and the application of concurrency is entirely appropriate.” 22/9/2021 11:55 AM

# Assignment 5

Implement the next release of your term project, employing *concurrency* if concurrency can fit; otherwise create a different project with concurrency. The same instructions as before apply to this completed Word document, the gray text, the 5 page limit, appendices, JUnit tests, and a ReadMe file. As usual, copying someone else’s application without clear attribution is plagiarism, and will be subjected to the College’s academic conduct process.

## 5.1 SUMMARY DESCRIPTION

*Evaluation criterion (i) applies*

One- or two-paragraph overall description of your proposed term project. Color red the parts changed from Assignment 2, if any (all in red if this is a separate application).

Your response replaces this.

## 5.2 ADDITIONAL REQUIREMENTS (FEATURES) IMPLEMENTED IN THIS RELEASE

*Evaluation criterion (i) applies*

Title and one or two sentences per requirement. Don’t repeat requirements implemented for prior assignments unless they are necessary to provide context—in which case, make it clear they are old.

### 5.2.1 Your title replaces this. (NEW/OLD)

Your response replaces this.

### 5.2.2 Your title replaces this. (NEW/OLD)

Your response replaces this.

### 5.2…. more as needed

## 5.3 I/O SUPPORTING THE NEW REQUIREMENTS LISTED ABOVE

*Evaluation criterion (ii) applies*

Provide an example of input / output showing the new features of your application.

### Input

Your response replaces this.

### Input / Output

Your response replaces this.

## 5.4 YOUR DIRECTORY

Show a screenshot of your directory. Include your “.dat” files (where objects are written). This should include JUnit tests—class-by-class, and method-by-method, except for trivial and inappropriate ones.

Your response replaces this.

## 5.5 DESIGN THIS

*Evaluation criterion (i) applies*

Supply a main use case, the class model, and the sequence diagram corresponding to the use case. These should be consistent. Indicate in red your class model where you applied object read, object write, streams and lambdas. Excellent assignments will typically include the use of Java FX (speak to your facilitator first if you wish to use alternative API’s) and event-driven programming.

Your response replaces this.

## 5.6 CODE SNIPPETS

*Evaluation criterion (iii) applies*

### 5.6.1 Code showing where concurrency is *defined THIS*

ReadFromLibraryJSON is a nested class within gui.ReadSequencePage.java

*/\*\*  
 \** ***@author*** *Michael Kramer  
 \* <p>  
 \* CS622 Spring 1, 2022 Advanced Programming Techniques  
 \* <p>  
 \* The purpose of this class is to provide a threaded approach to reading  
 \* files from a common library of shared files, not intended to be edited  
 \* by the current user, only read, and could be read by many users at once  
 \* at scale with ease  
 \*/*public class ReadFromLibraryJSON implements Runnable {  
 @Override  
 public void run() {  
 ComboBox filesInPubLib = getFilesInPubLib();  
 String filePath = filesInPubLib.getValue().toString();  
 if (filePath != "" && filePath != null) {  
 ReadFromJSON reader = getController().getReader();  
 ChordReadGUIController controller = getController();  
 Text display = getDisplay();  
 File path = new File(filePath);  
 File fileName = new File(path.getAbsolutePath().substring(path.getAbsolutePath().lastIndexOf("\\") + 1));  
 String nameNoExt = openFile(fileName);  
 ChordSequence seqFromLib = reader.readChordSequenceFromLibJSON(nameNoExt);  
 String analysis = controller.analyze(seqFromLib);  
 display.setText(analysis);  
 Platform.*runLater*(() ->  
 {  
 Label pubLibLabel = getPubLibLabel();  
 pubLibLabel.setText("\"" + nameNoExt + "\"" + " distinct chords on the console");  
 });  
 }  
 }  
}

### 5.6.2 Code showing where concurrency is *used THIS*

You’re The below code is within gui.ReadSequencePage.start() and illustrates creating a new cached thread pool from the ExecutorService class in order to ready a file in a library common to every user (simulated for now in project\_root/data/concurrencyLib directory)

*/\*\*  
 \* The purpose of this method is to read a file from a library of common  
 \* chord progressions and display the analysis for study  
 \*/*readPubFile.setOnAction(event ->  
{  
 ExecutorService executorService = Executors.*newCachedThreadPool*();  
 executorService.execute(new ReadFromLibraryJSON());  
 executorService.shutdown();  
 try {  
 executorService.awaitTermination(1, TimeUnit.*MILLISECONDS*);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
});

. User runs gui.ReadSequencePage.java and has a Public Library Option

Graphical user interface

Description automatically generated with low confidence

. User selects file from a prepopulated path to the concurrencyLib directory (Shared Library Simulation)

Text, letter

Description automatically generated

. User selects "Read Library Selection" and Java's ExecutorService executes the nested class ReadFromLibraryJSON.run() in a unique thread as multiple users may be reading from the same file at scale

Text, letter

Description automatically generated

## 5.7 YOUR CODE

*Evaluation criterion (iii) applies*

Unless your facilitator arranges another method, copy your Eclipse project to your file system, zip it, and attach it to your Blackboard response. Please contact your facilitator in advance if you require an exception.

Your response replaces this.

## 5.8 Evaluation

