Please replace this paragraph with your Assignment 1, including grading comments. Your term project will form a single *Word* document as the term progresses. Using *Document Map*, this is easy to navigate. --1/20/2021 9:07 AM

# Assignment 2

Implement the next release of your term project. You will incorporate exception handling and file I/O, as specified below.

Submit this completed Word document, including your name within the file name, observing and retaining the gray text like this. Retain the headings or edit them as indicated. Your Assignment 2 materials—in black 12-point Times New Roman—should not exceed 5 pages excluding references, figures, and appendices. Use the Appendix sections for additional material if you need to. These will be read on an as-needed basis.

As you code, use JUnit tests—package-by-package, class-by-class, and method-by-method, except for trivial ones and ones requiring I/O that cannot be unit tested. Use non-Junit classes for testing the latter, as in the worked example.

Include a ReadMe file describing where to run the application from, and including notes as necessary (not more).

Keep the evaluation criteria in mind, listed at the end and referred to in the headings.

## 2.1 SUMMARY DESCRIPTION, UPDATED AS APPLICABLE THIS

## *EVALUATION CRITERION (i) APPLIES*

One- or two-paragraph overall description of your whole proposed term project. Edit your last description as needed.

Your response replaces this.

## 2.2 I/O EXAMPLE FROM PROJECTED COMPLETED PROJECT, UPDATED AS APPLICABLE *EVALUATION CRITERION (i) APPLIES*

Provide an example of projected *concrete* output for example input, indicating how users will interact with your application. You will not be held to fulfilling exactly this—it intended to help us understand the probable direction of your application as a whole.

Your response replaces this.

## 2.3 REQUIREMENTS IMPLEMENTED IN THIS RELEASE NOT IMPLEMENTED BEFORE *EVALUATION CRITERION (i) APPLIES*

Your content response replaces this

### 2.3.1 Your title replaces this THIS

Your content response replaces this

### 2.3.2 Your title replaces this

Your content response replaces this.

### 2.3.3 Your title replaces this

### Your content response replaces this

### 2.3.4 Your title replaces this

### Your content response replaces this

### 2.3.5 Your title replaces this (add more requirements as needed)

Your content response replaces this

## 2.4 I/O EVIDENCE THAT THE ABOVE FUNCTIONALITY WAS ACHIEVED

## *EVALUATION CRITERION (ii) APPLIES*

## This typically consists of screen shots of input and output, together with text explaining their context.

Your response replaces this.

## 2.5 YOUR DIRECTORY

Show a screenshot of your directory. This should include a parallel directory of JUnit tests—package-by-package, class-by-class, and method-by-method, except for trivial ones.

Your response replaces this.

## 2.6 TECHNIQUES IMPLEMENTED *EVALUATION CRITERION (iii) APPLIES*

Your implementation should exploit *file IO* and *exceptions* at least once, in as natural a manner as possible. Using the headings below, explain where and how you applied these.

### 2.6.1 Class model and Sequence Diagram

Indicate clearly in your class model where you applied file IO and exception handling, including a user-defined exception if possible. “Enforce what you intend.” For example, make classes and members *static* or not as per their intended usage. To do this use tools, PowerPoint, or combine models as in [this RUML example](https://docs.google.com/spreadsheets/d/1vBmDVtWWh3EX0oehFFLRU0P6eR-fn4d0qVg1-XOUooM/edit?usp=sharing) (which you are free to copy, cut and paste from). Insert indications in red (as in the example) to show where the three features below apply.

Your response replaces this.

### 2.6.2 Code showing *file I/O THIS*

Show the relevant code (only). It should be clear where the code is located (class and method). Specify nontrivial methods with pre- and postconditions (and examples if this clarifies).

Your response replaces this.

### 2.6.3 Code showing *exception*, preferably a user-defined exception

Show the relevant code (only) and explain why *exceptions* are appropriate here. It should be clear where the code is located (class and method). Specify nontrivial methods with pre- and postconditions (and examples if this clarifies).

Your response replaces this.

## 2.7 YOUR CODE

Unless your facilitator requests or allows another method, copy your Eclipse project to your file system, zip it, and attach it. Please contact your facilitator in advance if you want to request an exception. Specify nontrivial methods with pre- and postconditions (and examples if this clarifies). For excellent work, specify the class invariants.

<Your response here>

## 2.8 EVALUATION OF ASSIGNMENT 2



## Appendix 1 (will be read as-needed only—add more as necessary)