# Assignment 3 5/23/25

Implement the next release of your term project. **You will incorporate Java generics and file I/O in a fitting manner (i.e., where they are needed), as specified below.**

Leverage an AI generator such as ChatGPT as much as you can to **continue creating** a real-world application. As described in the evaluation criteria below, your work will be assessed in terms of *your value added* (not simply on AI-generated material). Your value added consists of your choice of prompts together with your edits and additions to AI-generated material that result in capable and high quality code. Show your value added in red font and by means of explanations. For figures, insert comments (in red) that describe clearly your value added.

Please provide all code in text format, not in screenshots, so you can highlight in red your value added. If you performed significant prompt work, please note this in the relevant sections with added explanations. **To get credit for your prompt work, comment on what you consider significant prompt contributions.**

Accompany code and diagrams with explanations.

For functions, use the functionName(arguments) / INTENT / EXAMPLE / DEFINITIONS / PRECONDITIONS / POSTCONDITIONS format.

Your application must provide an interactive input mechanism, commonly a CLI or GUI. The user must be able to supply different values and responses without recompiling, relaunching, or editing source code.

Submit this completed Word document. Insert your material as indicated. Please observe and retain the gray text. Your materials—in black 12-point Times New Roman—should not exceed 5 pages excluding the gray instructions, references, figures, and appendices. Use the Appendix sections for additional material if you need to and refer to them in the document body. These will be read only on an as-needed basis.

Please develop in Eclipse—preferably—or else IntelliJ (talk to your facilitator about exceptions). As you code, use JUnit tests whenever possible—package-by-package, class-by-class, and method-by-method, except for trivial methods and those requiring I/O. Use testing classes for testing the latter. Keep the evaluation criteria in mind, listed at the end—**point out your contributions with these in mind.**

Housekeeping:

1. Include a ReadMe file that contains necessary execution notes and describing where to run the application from. All JUnit tests will be assumed runnable.
2. After you have completed the questions, make sure you have saved the file.
3. Please save this completed document with the file name: METCS622\_Assignment1\_FirstnameLastname.
4. To upload the completed Draft Assignment 1, click the "Browse My Computer" to upload your Word file, and then click "Submit".
5. Export your project from your IDE using its export feature and provide it as a second attachment.

## 1 SUMMARY DESCRIPTION, UPDATED AS NECESSARY

One- or two-paragraph overall description of your proposed term project. Color in red the parts different from Assignment 2.

Your response replaces this.

## 2 ADDITIONAL REQUIREMENTS IMPLEMENTED IN THIS RELEASE

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

### 2.1 Your title replaces this. (OLD / NEW REQUIREMENT)

Your response replaces this.

### 2.2 Your title replaces this. (OLD / NEW REQUIREMENT)

Your response replaces this.

### 2.3 …

## 3 I/O EVIDENCE OF ACCOMPLISHING THE REQUIREMENTS LISTED ABOVE

Provide examples and explanations of actual input / output corresponding to the requirements above. **Include enough scenarios to make your accomplishment of the requirements entirely clear.**

Your response replaces this.

### Input File(s)

Your response replaces this.

### Console or GUI Input / Output

Your response replaces this.

### Output File(s) (if applicable)

Your response replaces this.

## 4 YOUR DIRECTORY

Continue to divide your code into well-named packages, each containing a singleton Facade object. If the package is named my.package, the Façade object should be named FacadeMyPackage. Obtaining the singleton object should be done with getTheInstance(). Access to functionality within each my.package should be only via myPackageAccess().

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

A screenshot of your directory replaces this.

## 5 YOUR UPDATED CLASS MODEL AND CLARIFICATION OF HOW THE EXECUTION WORKS

Supply a main use case, the class model, and the sequence diagram corresponding to the use case. These should be consistent and clear. Indicate clearly in your class model where you applied generics. To do this use tools, PowerPoint, or a combine models as in [this example](https://docs.google.com/spreadsheets/d/1vBmDVtWWh3EX0oehFFLRU0P6eR-fn4d0qVg1-XOUooM/edit?usp=sharing). Insert indications in red to show where generics apply.

Your response replaces this.

## 6 WHERE GENERICS ARE IMPLEMENTED

### 6.1 Class model fragment showing generic class(es)

Explain where and how you applied *generic classes* in your class model.

Your response replaces this.

### 6.2 Code (including test code), input (if applicable), and output showing generics and collections

Include at least two uses of Collection classes.

Your response replaces this.

### 6.3 Explanation of generics use

Explain why your use of *generics* is appropriate here. Explain what the class model and code would be without it.

Your response replaces this.

## 7 YOUR CODE

Unless your facilitator arranges 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 another transmission process (e.g., github).

## 8 INSTRUCTOR’S EVALUATION



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