Department of Electrical and Software Engineering
Schulich School of Engineering

ENSF 694 - Principles of Software Development II
Summer 2023

Project Specifications

Objectives:

• Create a library for common data structures

1. Searching and Sorting
    a. Interpolation Search
    b. Only one from Selection Sort, Quicksort, Insertion Sort, Merge Sort, Bubble Sort
2. Linear data structures
    a. Singly Linked List
    b. Doubly Linked List
    c. Stack
    d. Queue
3. Tree structures
    a. Binary Search Trees
4. Graph algorithms
    a. Breadth First Search
    b. Depth First Search
    c. Dijkstra

Submission Type: Individual
Submission Deadline: August 08, Tuesday, 11:59 PM
Submission Mode: Github link and the 'myLibrary.jar' file in d2l 'Project' dropbox, push the codes in the github classroom

Go to this link - https://classroom.github.com/a/0cK0X6y3
Refresh and accept the Project link
Clone the repository and then push your project code folder
Then submit the github link and the jar file to the d2l dropbox 'Project'

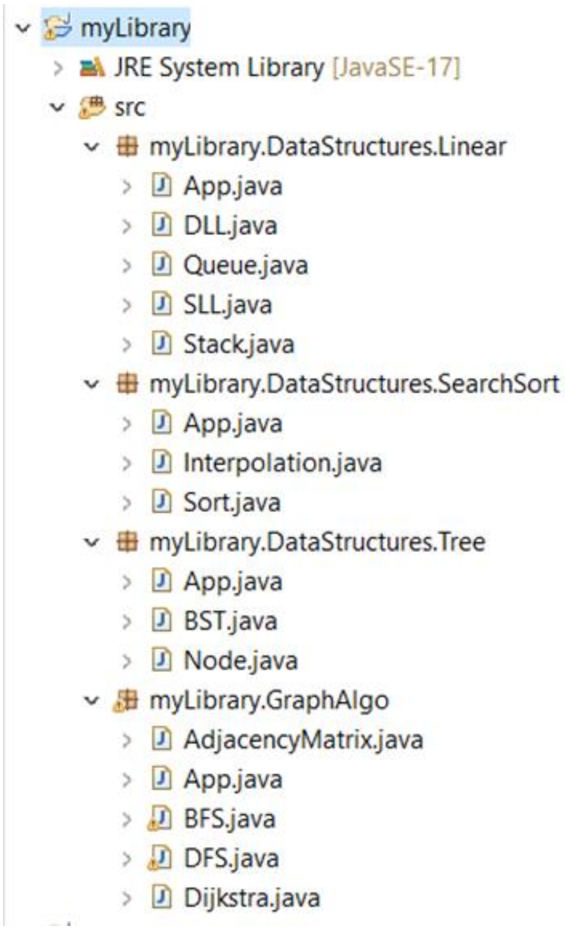
Weight: Total project weight is 30% of the course. The project components will be divided between the labs and final submission.
The problems solved in the labs will be using some of the algorithms from the above list. You can modify those according to the project requirements and add the modified codes in the project implementation.
7 labs – 5%
Final Project – 25%

Project outline:
In this project, you will develop your own library of data structures with their associated algorithms. The library will have multilevel structure to separate each group of data structures into groups based on their class. The library will be created with a package name "myLibrary" and will have the following underlying structure:

Your library will split into 2 main subfolders -

1. DataStructures – this will contain subfolders for each class of data structures you will be implementing mentioned below
   - SearchSort
     - Interpolation (java file)
     - Sort (java file)
   - Linear
     - SLL (java file)
     - DLL (java file)
     - Stack (java file)
     - Queue (java file)
   - Tree
     - BST (java file)
2. GraphAlgo - this will contain subfolders for each class of data structures you will be implementing mentioned below
   - BFS (java file)
   - DFS (java file)
   - Dijkstra (java file)

The structure will look like this-



```
myLibrary
  JRE System Library [JavaSE-17]
  src
    myLibrary.DataStructures.Linear
      App.java
      DLL.java
      Queue.java
      SLL.java
      Stack.java
    myLibrary.DataStructures.SearchSort
      App.java
      Interpolation.java
      Sort.java
    myLibrary.DataStructures.Tree
      App.java
      BST.java
      Node.java
    myLibrary.GraphAlgo
      AdjacencyMatrix.java
      App.java
      BFS.java
      DFS.java
      Dijkstra.java
```

NOTES:

- All data structures will be implemented for integer data types as the data member.
- You should not use the java.util.LinkedList, java.util.stack, java.util.queue, or any of their built in methods for creating and accessing your linked lists, stacks, or queues.
- Using java.util.Arrays is okay.
- Follow the format from “Project -> myLibrary.zip”. More details instructions are available there.
- App.java in each folder handles inputs, outputs, and calls the methods from other classes of that folder.
- Do not change anything in App.java files and follow the exact structure mentioned in the uploaded project. The only exception is the X_Sort(). As you can choose one of the mentioned sorting algorithms, you can replace X_Sort() with your sorting name and arguments.
- Do the sorting for interpolation in Interpolation.java. It is okay if the original search key index doesn’t match the final index after sorting.
- For the graph part, the randomly generated graph might be disconnected. So, you might have incomplete paths for the traversals.
- You can add more classes to the folders if needed.
- You will convert the project into a Maven project (or you can start with a Maven project) and create a .jar file named ‘myLibrary.jar’ at the end.
- You can include your own unit testing, but make sure to include sufficient comments to clarify your steps.

Project finalization:
To finalize and complete your project, you will build the project using maven to generate a myLibrary.jar file. This will be your compiled library that you can copy into any project and import it to use your data structure implementations whenever needed.

The TAs will be testing your Project by Importing your library jar file into a tester program to run some tests so it is very important that you adhere to naming conventions that will be published in each module separately.

Sample Input-Outputs for each part: ‘Project -> Sample_Screenshots.zip’