EECS 3311
Software Design
Lab 1, practice on OOP

This lab is intended to help you get familiar with the basic OOP design principles. This lab will not be graded, it's a practice lab.

Check the Amendments section of this document regularly for changes, fixes, and clarifications.

Ask questions on the course forum on the Slack workspace.

1 Policies

- Your (submitted or un-submitted) solution to this assignment (which is not revealed to the public) remains the property of the EECS department. Do not distribute or share your code in any public media (e.g., a non-private Github repository) in any way, shape, or form before you get the permission from your instructors.

Amendments

- so far so good

2 Problem

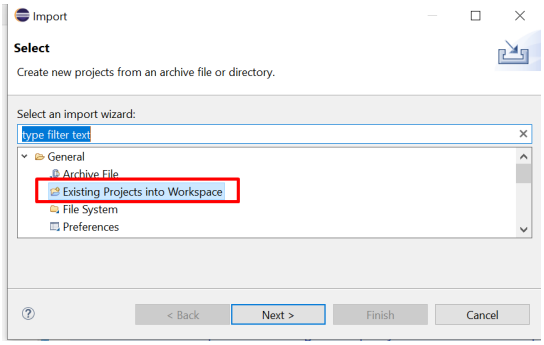
In this lab, you are expected to tackle the problem of sorting **HashMap <K, V>** data. HashMap provides default sorting algorithm based on keys (i.e., K), however it does not enable sorting the data based on values (i.e., V). There are many different scenarios that we need to sort data in a HashMap based on its values, e.g., HashMap data structure are widely used to store students' IDs (keys) and their GPAs (values). When we rank students by their GPAs, we need the value-based HashMap sorting functionality.

There are many different sorting algorithms [1], in this lab, we will adapt two basic sorting algorithms for sorting HashMap based on the values **ascendingly**, i.e., **Bubble Sort**[1] and **Max Heap Sort**[2]. When sorting the values of Maps, we should also move the keys accordingly. You can use two instances of *ArrayList* to store the values and keys and swap elements in these two ArrayLists synchronously during the sorting tasks.

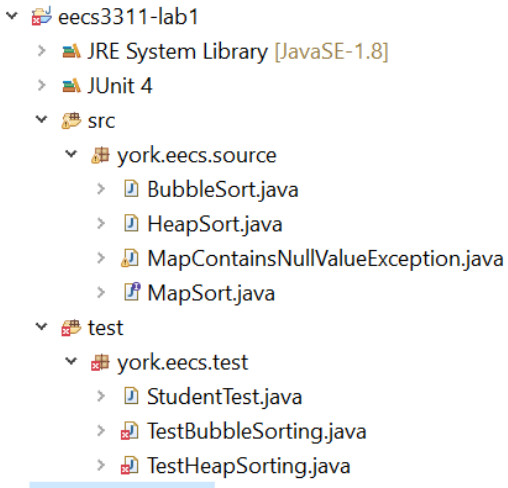
3 Getting Started

- Unzip the file, and you should see a directory named **eecs3311-lab1**. It's a Java project in Eclipse.
- You can import this project into your Eclipse as an General Java project.

[1] https://en.wikipedia.org/wiki/Bubble_sort
[2] https://en.wikipedia.org/wiki/Heapsort



- The start project should have the following structure and **on default does not compile.**



4 You Tasks

4.1 Complete the BubbleSort and HeapSort Classes

- You are expected to write valid implementations in the **BubbleSort** and **HeapSort** classes. Each instance of "TODO:" in these two classes indicates which implementation you have to complete.
- You are expected to implement the specified two sorting algorithms, replace the specified sorting algorithms with other sorting algorithms will receive penalty.
- Study the **TestBubbleSorting** and **TestHeapSorting** class carefully: it documents how **BubbleSort** and **HeapSort** expected to work.
- You must not change any of the methods, parameters, or statements in the **TestBubbleSorting** and **TestHeapSorting** classes.

4.2 Write Your Own Test Cases

In the **StudentTest** class, you are required to add as many tests as you judge necessary to test the correctness of **BubbleSort** and **HeapSort**.

- You must add at least 10 test cases in **StudentTest**, and all of the must pass. (In fact, you should write as many as you think is necessary.)

- You will not be assessed by the quality or completeness of your tests (i.e., we will only check that you have at least 10 tests and all of them pass). However, write tests for yourself so that your software will pass all tests that we run to assess your code.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.