EECS 3311
Software Design
Lab 1 (100 points), Version 1

Instructor: Song Wang
Release Date: Jan 20, 2020

Due: 11:59 PM, Wednesday, Feb 3, 2020

All your lab submissions must be compilable on the department machines. It is then crucial that should you choose to work on your own machine, you are responsible for testing your project before submitting it for grading. This lab is intended to help you get familiar with the basic OOP design principles.

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

Ask questions on the course forum on the eClass site.

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.

- You are required to **work on your own for this lab**. No group partners are allowed.
- When you submit your solution, you claim that it is solely your work. Therefore, it is considered as an violation of academic integrity if you copy or share any parts of your code or documentation.
- When assessing your submission, the instructor and TA may examine your doc/code, and suspicious submissions will be reported to the department/faculty if necessary. We do not tolerate academic dishonesty, so please obey this policy strictly.

- You are entirely responsible for making your submission in time.

- You may submit multiple times prior to the deadline: only the last submission before the deadline will be graded.
- Practice submitting your project early even before it is in its final form.
- No excuses will be accepted for failing to submit shortly before the deadline.
- Back up your work periodically, so as to minimize the damage should any sort of computer failures occur. You can use a **private** Github repository for your labs/projects.
- The deadline is strict with no excuses.
- Emailing your solutions to the instruction or TAs will not be acceptable.

Amendments

- so far so good

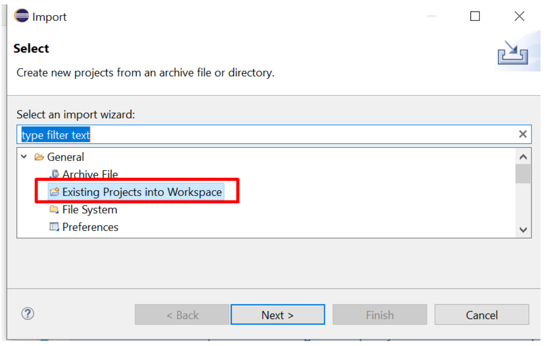
2 Problem

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.

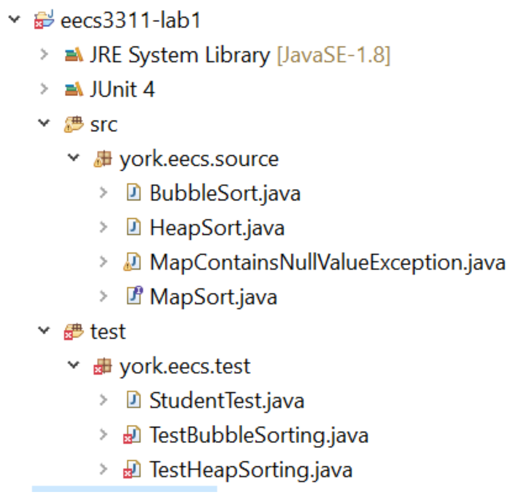
There are many different sorting algorithms [1], in this lab, we will adapt two basic sorting algorithms for sorting HashMap based on the values **ascendingly**, i.e., **Bubble Sort**[1] and **Max Heap Sort**[2]. When sorting the values of Maps, we should also move the keys accordingly. You can use two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

3 Getting Started

- Go to the course eClass page for Section Z. Under Lab 1, download the file **EECS3311_Lab1.zip** which contains the starter project for this lab.
- Unzip the file, and you should see a directory named **eecs3311-lab1**. It's a Eclipse project.
- You can import this project into your Eclipse as an General Java project.



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



```
eecs3311-lab1
    JRE System Library [JavaSE-1.8]
    JUnit 4
    src
        york.eecs.source
            BubbleSort.java
            HeapSort.java
            MapContainsNullValueException.java
            MapSort.java
    test
        york.eecs.test
            StudentTest.java
            TestBubbleSorting.java
            TestHeapSorting.java
```

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

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 **TestHeap-Sorting** 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.

5 Submission

To get ready to submit:

- Close Eclipse
- Zip your lab1 project with name 'EECS3311_Lab1.zip'.

By the due date, submit via the following command:

```
submit 3311 lab1 EECS3311_Lab1.zip
```

References

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