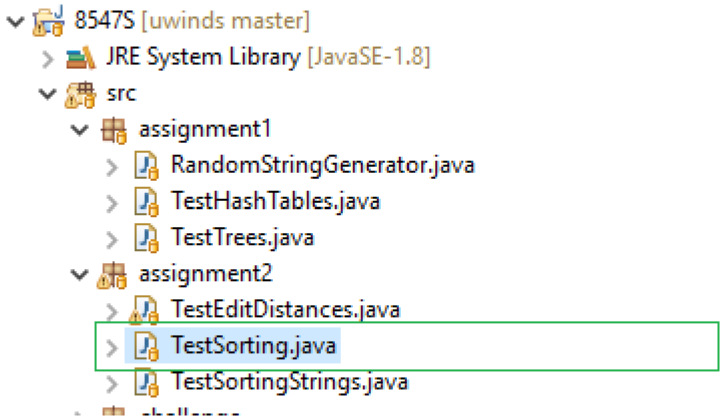
Assignment 2

As a student at the University of Windsor, I pledge to pursue all endeavours with honour and integrity and will not tolerate or engage in academic or personal dishonesty. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work.

Wen Dong #110057395

Task #1 and #2

1. The source code



```
8547S [uwinds master]
  JRE System Library [JavaSE-1.8]
  src
    assignment1
      RandomStringGenerator.java
      TestHashTables.java
      TestTrees.java
    assignment2
      TestEditDistances.java
      TestSorting.java
      TestSortingStrings.java
```

2. Output of the java file looks as below:

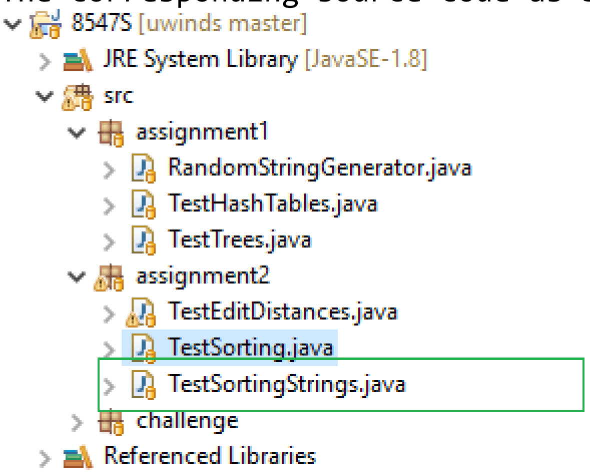
```
Testing sorting 100,000 random numbers for 100 times:
Algorithsm   Avg time(ms)
Mergesort    71
Quicksort    50
Heapsort     75
dual-pivot   70
```

3. Comments:

From the results, we can see all the four sort algorithms perform at a same decent time complexity level, compliant with the time complexity we learned form class, which is O(n log n)

Task #3 and #4

1. The corresponding source code as below



```
8547S [uwinds master]
  JRE System Library [JavaSE-1.8]
  src
    assignment1
      RandomStringGenerator.java
      TestHashTables.java
      TestTrees.java
    assignment2
      TestEditDistances.java
      TestSorting.java
      TestSortingStrings.java
    challenge
  Referenced Libraries
```

2. Output of the java file looks like the table below:

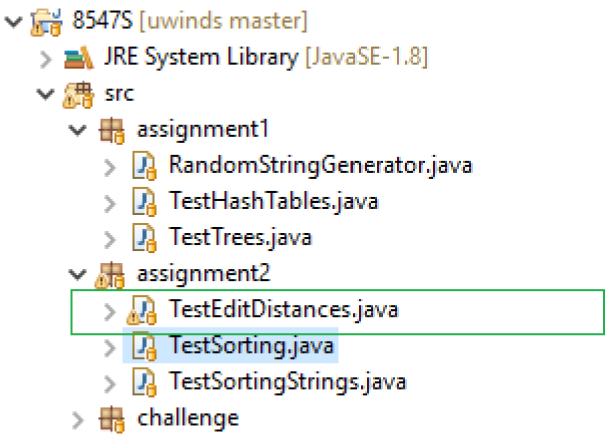
```
Testing sorting 100000 random strings for 100 times:
Avg time(ms) |      4      6      8      10
------------------------------------------------------
Mergesort    |      148    134    131    126
Quicksort    |      104    101    101    106
Heapsort     |      175    186    220    190
dual-pivot   |      109    117    129    121
Radixsort    |      96     126    233    212
```

It shows each algorithm's average time of sorting 100000 random strings respectively of length 4, 6, 8, and 10.

3. Comment:
   According to the results, I will choose Radix sort for sorting strings of small length (<=4), as its performance excels in that case. On the other hand, for lengthy strings sorting I prefer Quick sort and Dual-pivot quick sort, as they outperform in that scenario, moreover, they conduct the sorting process in place while Merge sort requires additional space.

Task #5

1. The corresponding source code as below



```
8547S [uwinds master]
  JRE System Library [JavaSE-1.8]
  src
    assignment1
      RandomStringGenerator.java
      TestHashTables.java
      TestTrees.java
    assignment2
      TestEditDistances.java
      TestSorting.java
      TestSortingStrings.java
    challenge
```

2. Output of the java file looks like the table below:

```
Calculating Edit Distance for 1000 pairs of random strings:
Pairs  StrLen Avg time(ns)
1000   10     18275
1000   20     43044
1000   50     63136
1000   100    168248
```

3. Comments:

The implementation of the Edit Distance solution is of time complexity O(nm) according to what we learnt from the class, and the testing results can roughly reflect that, it will be more clear if we increase the testing times to a larger number, like 100,000

```
Calculating Edit Distance for 100000 pairs of random strings:
Pairs  StrLen Avg time(ns)
100000 10     3715
100000 20     9052
100000 50     35039
100000 100    141366
```