

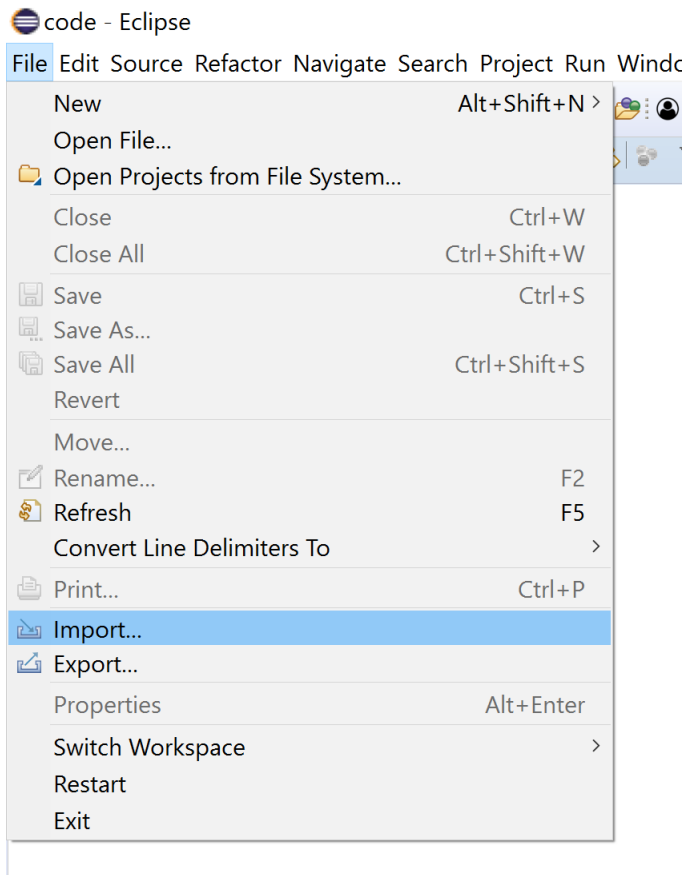
Getting Started with Assignment 1b

COMP 550.001 Fall 2017

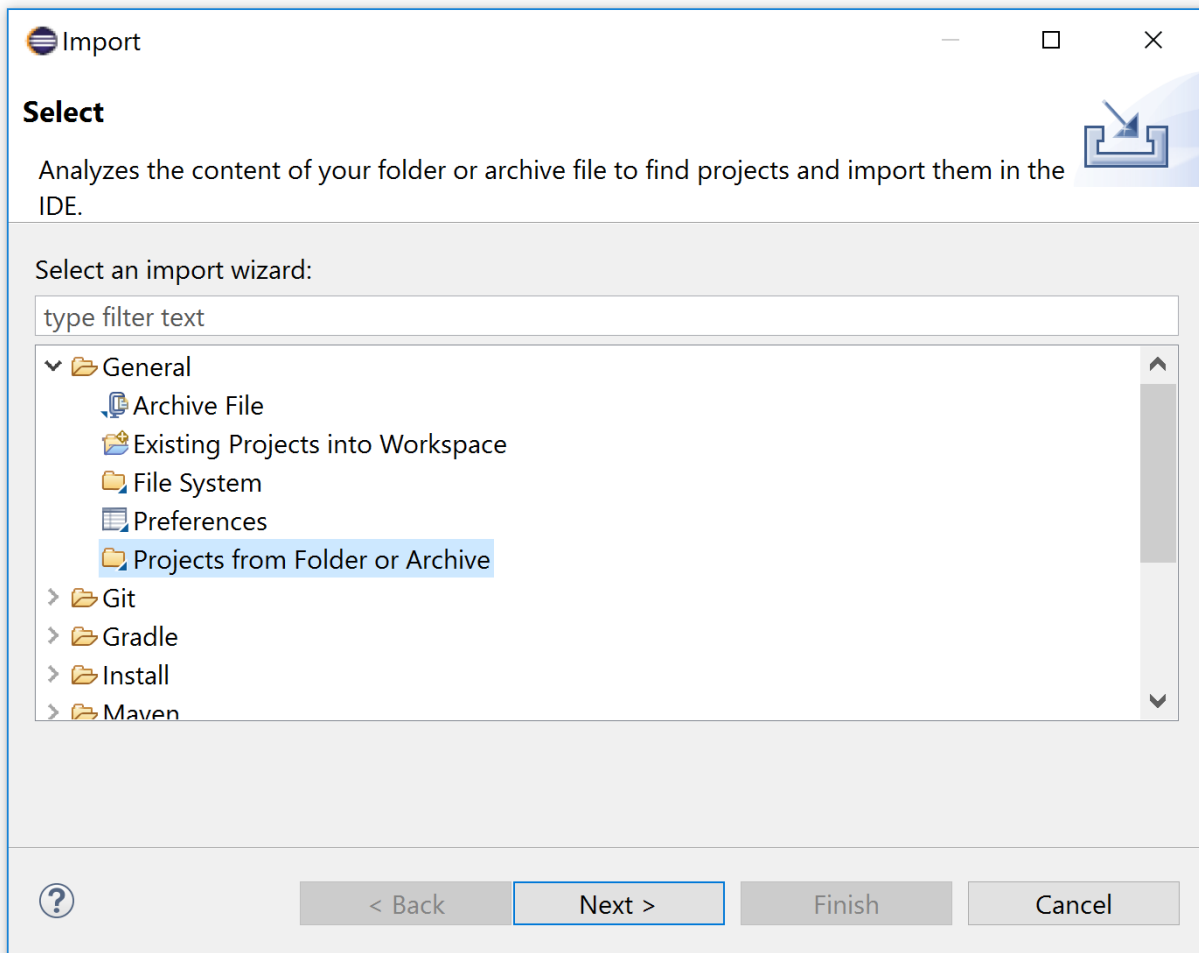
First, download the .zip file from <http://cs.unc.edu/~tamert/comp550-f17/hw1.zip>. This tutorial will use the Eclipse editor to import and run the starter code.

Import the .zip file as an Eclipse project

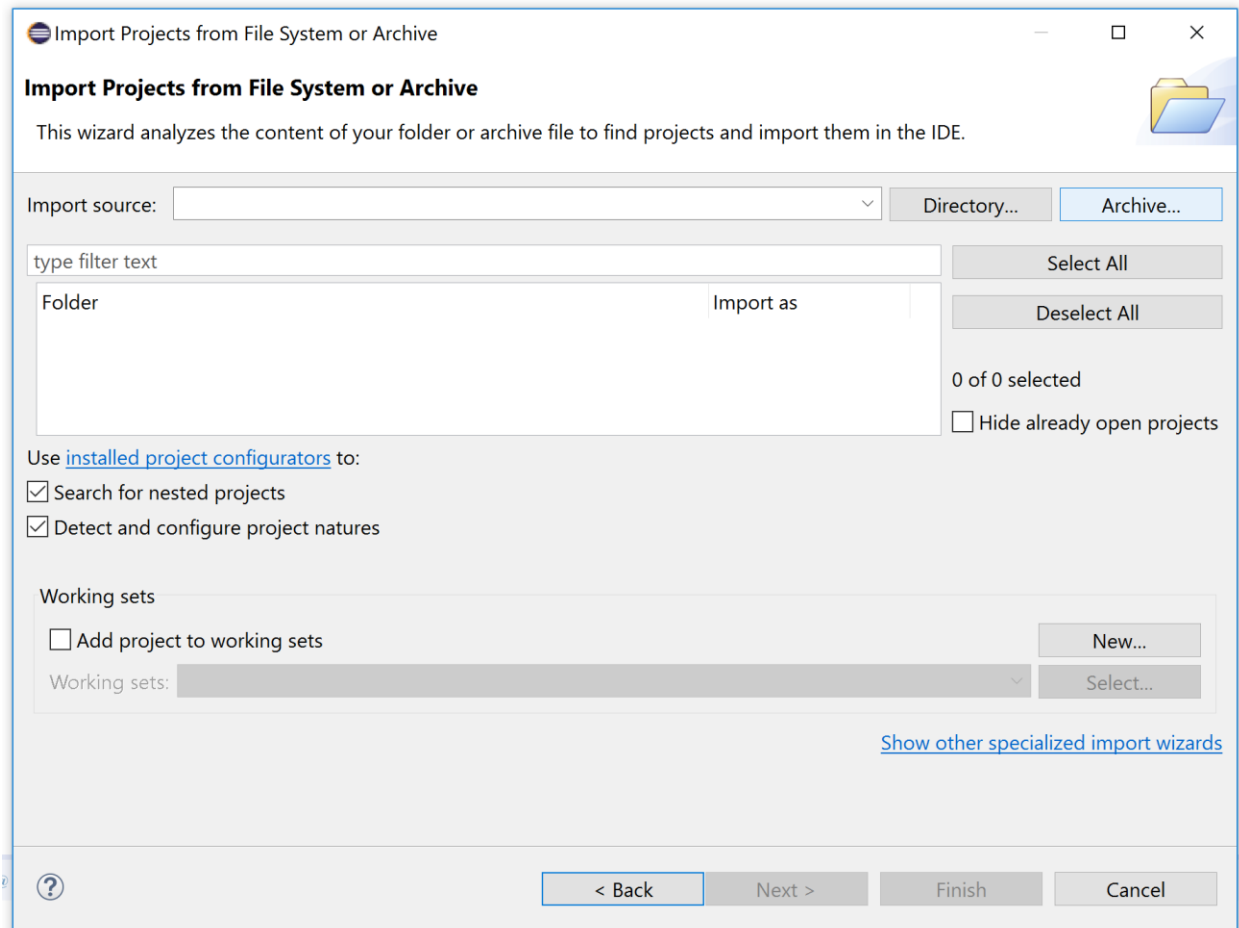
First, go to “File”->“Import”.



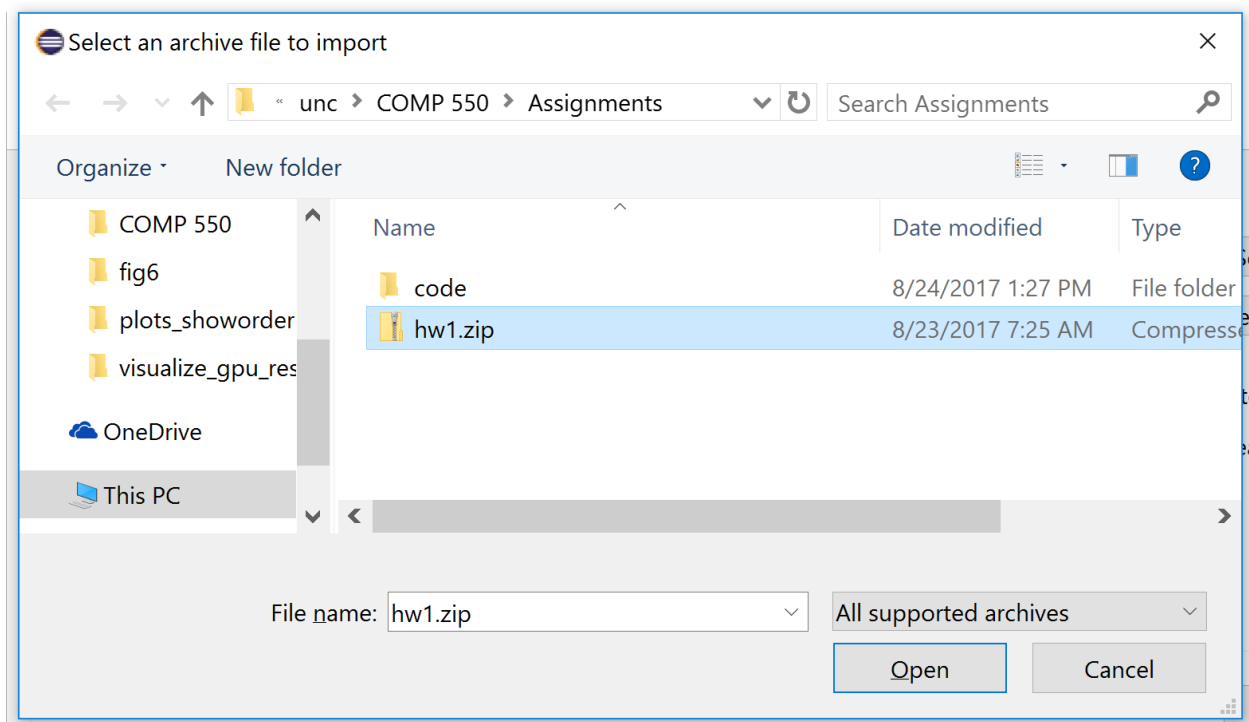
In the Import dialog, select “General”->“Projects from Folder or Archive” and click “Next >”.



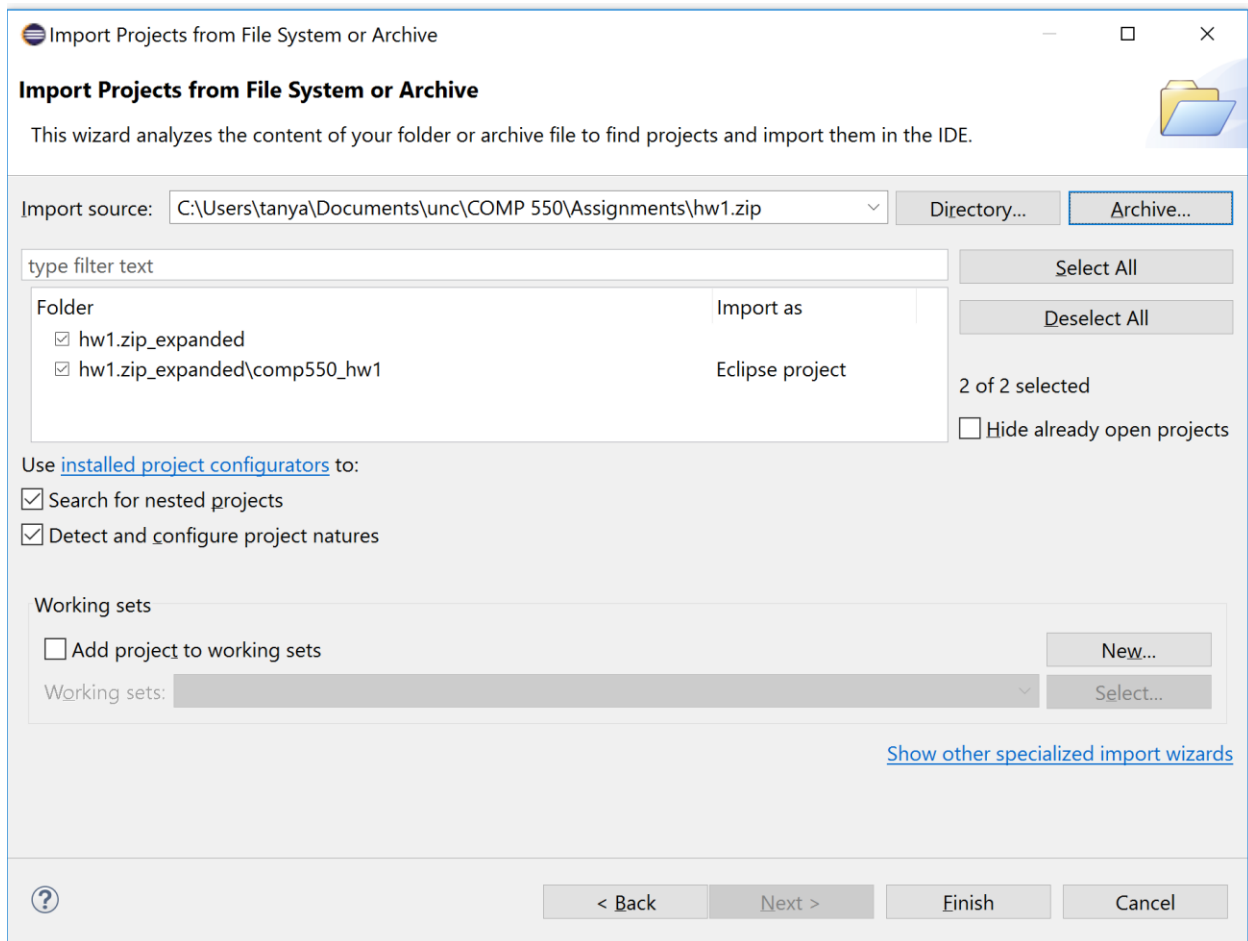
Next, select the “Archive...” button.



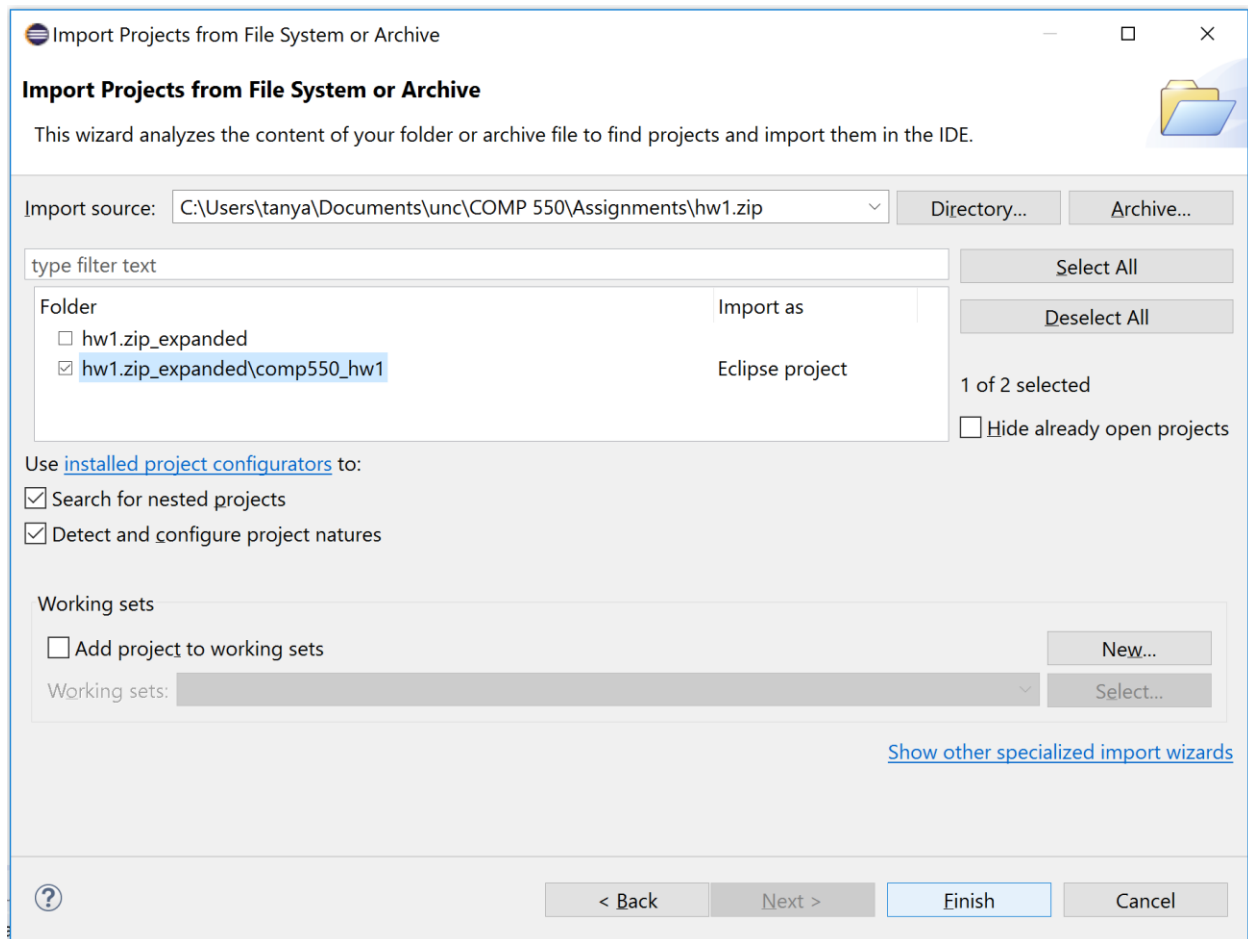
Navigate to where you saved the hw1.zip file, select the file, and click “Open”.



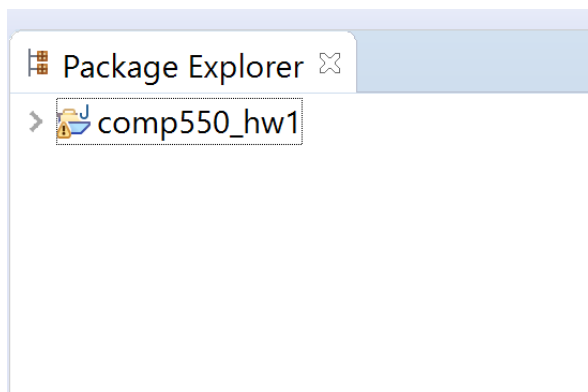
The Import dialog should now look like the following:



You only need the Eclipse project, so uncheck “hw1.zip_expanded” and click “Finish”.



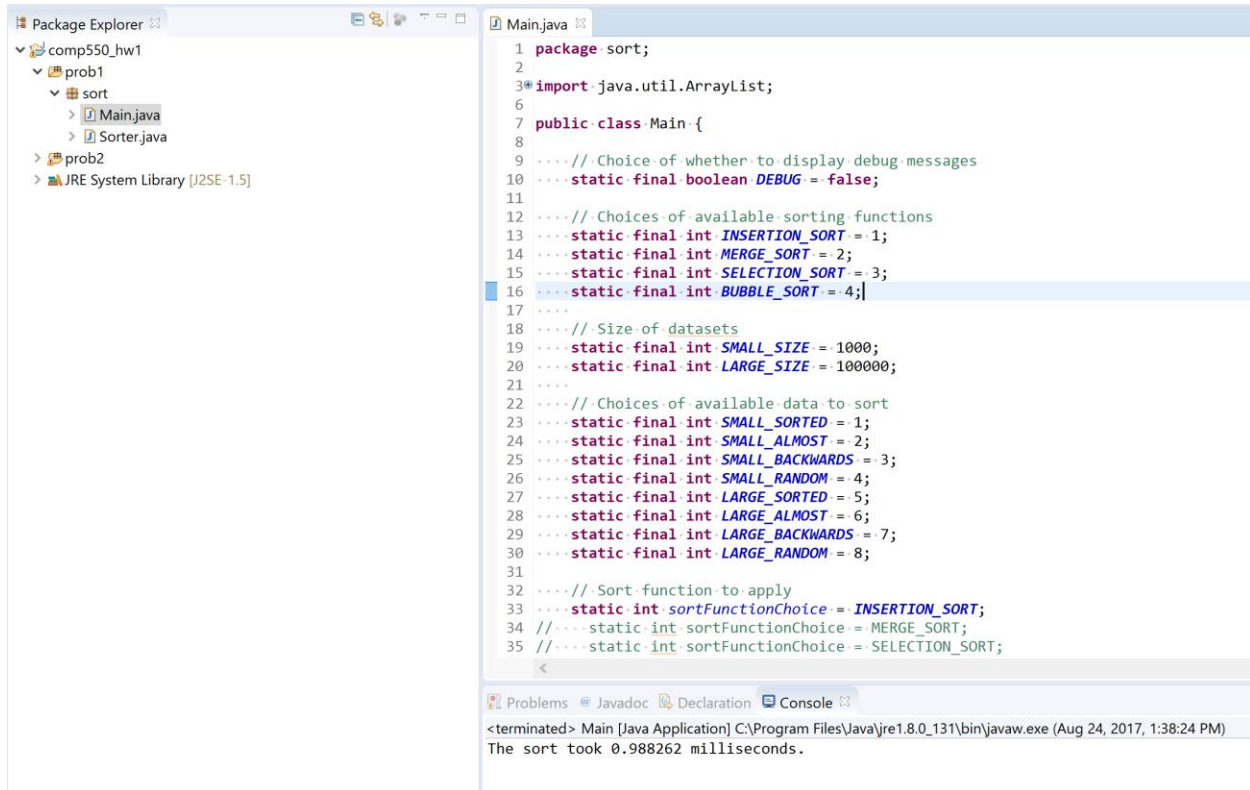
The project comp550_hw1 should now be listed in the Package Explorer.



Run the starter code

First, open prob1/sort/Main.java and run it. On success, it prints a message of the form:

“The sort took x.xxxxxx milliseconds.”



The screenshot shows an IDE with the Package Explorer on the left and the Main.java file open in the editor. The Package Explorer shows a project named 'comp550_hw1' with a sub-project 'prob1' containing a 'sort' package. Inside the 'sort' package, there are two files: 'Main.java' and 'Sorter.java'. The 'Main.java' file is selected. The editor shows the following code:

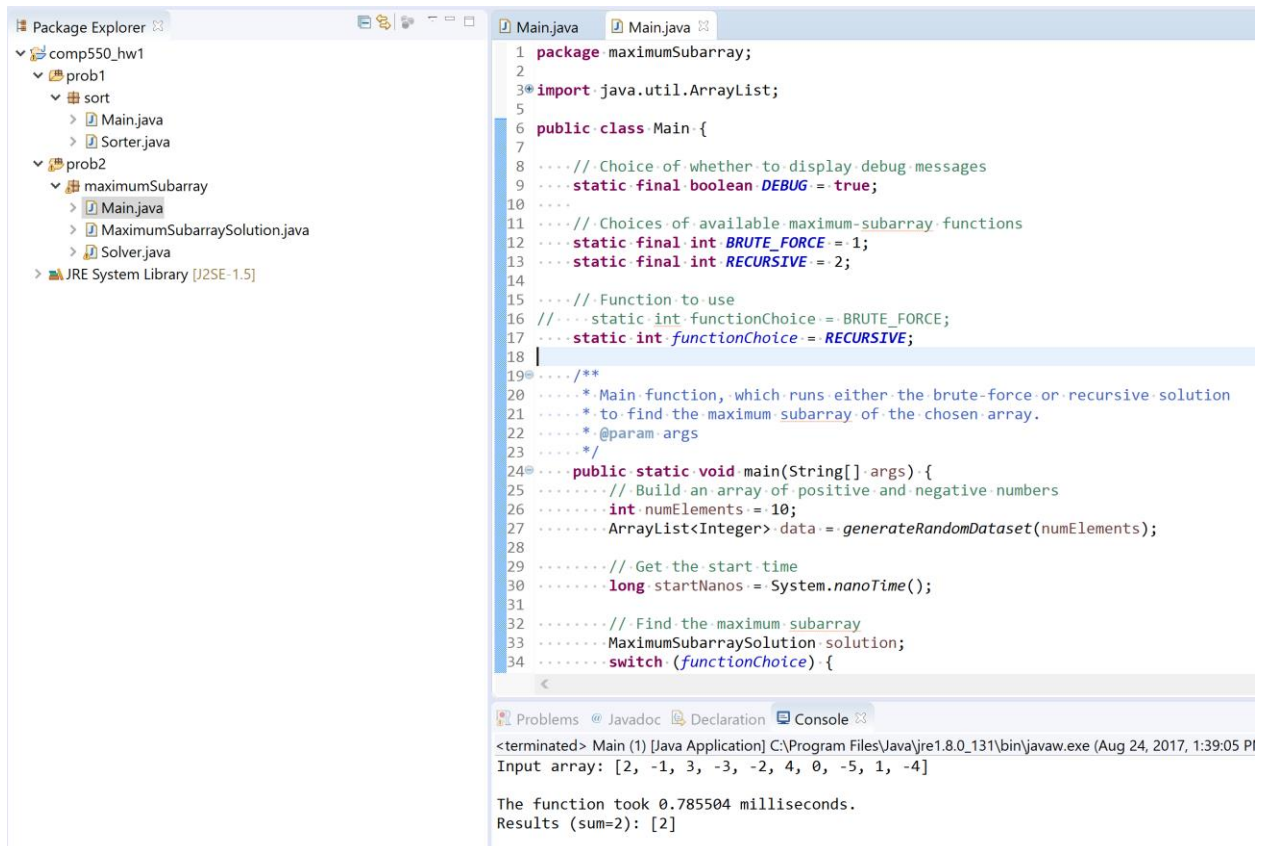
```
1 package sort;
2
3 import java.util.ArrayList;
4
5 public class Main {
6
7     // Choice of whether to display debug messages
8     static final boolean DEBUG = false;
9
10    // Choices of available sorting functions
11
12    static final int INSERTION_SORT = 1;
13    static final int MERGE_SORT = 2;
14    static final int SELECTION_SORT = 3;
15    static final int BUBBLE_SORT = 4;
16
17    // Size of datasets
18    static final int SMALL_SIZE = 1000;
19    static final int LARGE_SIZE = 100000;
20
21    // Choices of available data to sort
22
23    static final int SMALL_SORTED = 1;
24    static final int SMALL_ALMOST = 2;
25    static final int SMALL_BACKWARDS = 3;
26    static final int SMALL_RANDOM = 4;
27    static final int LARGE_SORTED = 5;
28    static final int LARGE_ALMOST = 6;
29    static final int LARGE_BACKWARDS = 7;
30    static final int LARGE_RANDOM = 8;
31
32    // Sort function to apply
33    static int sortFunctionChoice = INSERTION_SORT;
34    // static int sortFunctionChoice = MERGE_SORT;
35    // static int sortFunctionChoice = SELECTION_SORT;
```

The console output at the bottom shows the following message:

```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Aug 24, 2017, 1:38:24 PM)
The sort took 0.988262 milliseconds.
```

Then, open prob2/maximumSubarray/Main.java and run it. On success, it prints out debug messages and a message of the form:

“The function took x.xxxxxx milliseconds.”



```
1 package maximumSubarray;
2
3 import java.util.ArrayList;
4
5
6 public class Main {
7
8     // Choice of whether to display debug messages
9     static final boolean DEBUG = true;
10
11     // Choices of available maximum-subarray functions
12     static final int BRUTE_FORCE = 1;
13     static final int RECURSIVE = 2;
14
15     // Function to use
16     static int functionChoice = BRUTE_FORCE;
17     static int functionChoice = RECURSIVE;
18
19     /**
20      * Main function, which runs either the brute-force or recursive solution
21      * to find the maximum subarray of the chosen array.
22      * @param args
23      */
24     public static void main(String[] args) {
25         // Build an array of positive and negative numbers
26         int numElements = 10;
27         ArrayList<Integer> data = generateRandomDataset(numElements);
28
29         // Get the start time
30         long startNanos = System.nanoTime();
31
32         // Find the maximum subarray
33         MaximumSubarraySolution solution;
34         switch (functionChoice) {
```

<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Aug 24, 2017, 1:39:05 PM)
Input array: [2, -1, 3, -3, -2, 4, 0, -5, 1, -4]

The function took 0.785504 milliseconds.
Results (sum=2): [2]

Starter code structure

The starter code is broken into two separate folders and packages, one for each problem. You should read through the existing code to see how it works. Each Main.java file has a boolean DEBUG at the top that you can set to true to view additional output. In addition, both files select the algorithms to use via static ints defined before the main() function: sortFunctionChoice and datasetChoice for problem 1, and functionChoice for problem 2.

For problem 1, you should run each function-dataset combination to generate a table of timing results. You are welcome to modify existing code, but it should not be necessary.

For problem 2, you will have to fill in the function definitions of findMaximumSubarrayBruteForce() and findMaximumSubarrayRecursive() in Solver.java, and then run both with a range of input data sizes (numElements in main()) to generate a table of timing data.