University of New Brunswick
CS1083 – Open Access Version

Module 8 Assignment

The purpose of this assignment is to give you practice:

- converting the merge sort algorithm to work with Comparable; and
- demonstrating the time complexity of $O(N^2)$ versus $O(N \log_2 N)$.

For all assignments:

- Follow the instructions in the document "Java Coding Guidelines.pdf" available within the "Start Here" module, under the "Assignments" topic.
- For each .java file you create, include a javadoc comment (one that begins with /**) at the beginning of the file that describes that Java class. This comment block should include a line that begins with @author followed by your name and student number on the same line. (Note: inclusion of this comment is part of the instructions given in "Java Coding Guidelines.pdf")
- Include comments throughout your programs to explain any non-obvious portions of your code.
- It is recommended that you create a separate folder on your computer for each assignment. You might even wish to create separate sub-folders within an assignment folder if the assignment has multiple parts. Keeping your work organized makes it easier to find things later when you want to review what you have done.
- Few things in life are more frustrating than losing your work while working on an assignment. Get in the habit of saving frequently when working on an assignment. Also, regularly make backup copies of any files you create as part of your work for this course.
- Each module has a link to the discussion forum where you can ask questions and seek help in completing your assignments.
- Submitting your assignment involves creating a pdf file and uploading that single file to the assignment drop box on D2L. In general, that file will tend to include all Java code you wrote for the assignment, plus any output from running your programs that the assignment instructions specify you should capture. Specific submission instructions are included at the end of each assignment.
- To create the pdf file for each assignment, begin by opening a new document using the word processing program of your choice. Save the document using the name “Your Name CS1083 Module x Assignment Submission.docx” Replace x

with the correct module number. “docx” may be different depending on which word processing software you choose to use.

- At the beginning of your submission document enter your name, your student number, CS1083, the assignment name (this one is “Module 2 Assignment”), and the date. It doesn’t matter if the date is when you started working on the assignment or when you submit it – either will do.
- You can add content to your submission document as you work on the various questions in the assignment. Clearly label each part of the assignment (“Part A” etc.) in your document. When your document is complete, save / export as a pdf file. Always make sure your pdf file opens properly before uploading it.
- To include Java code in your submission document, copy all the text in your .java file and then paste that text into your submission document. Use a monospaced font (e.g.: Consolas, Courier) for your code to maintain proper indentation.
- To include output from running your program in your submission document, you have two choices. You can either (a) copy and paste the text from your command prompt window, or (b) capture a screen shot of the output and include that as a picture / image in your submission document. Either approach is fine.
- To copy text from your command prompt window, try selecting the desired text and then pressing either command-c (Mac) or control-c (Windows or Linux). If you have issues, you can always use Google to see how to do this on your specific type of computer.
- To capture a screen shot of a selected portion of your command prompt window, try command-shift-4 (Mac), WindowsKey-shift-s (Windows), or shift-PrtScrn (Linux).
- Once you have pasted your output text into your submission document, make sure it is in a monospaced font (e.g.: Consolas, Courier) so you retain the alignment of output.

Merge Sort for Comparable

Copy all the Java and spreadsheet files you worked with for the Module 4 Assignment into a new directory for this Module 8 Assignment.

Your task is to:

a) Add merge sort to the Sorter class; and
b) Extend the TimeTest class to test both the selection sort and the merge sort.

From MergeSortTest.java (available for download earlier in this module), copy the two mergeSort() methods and the merge() method into **Sorter.java**.

Those three methods are designed to sort an array of type int. The array is declared as a parameter for all three methods as:

```
int[] a
```

Update all three methods to change the type of this array to `SomeType`, which is the same type of array handled by the selectionSort() method already in the Sorter class.

Add a line similar to the following at the beginning of the convenience version of the mergeSort() method:

```
if (qty < 2 || qty > a.length)
   return;
```

This ensures that the merge sort code executes only when the value of `qty` makes sense.

The merge() method creates a temporary array `b` as follows:

```
int[] b = new int[n];
```

In this line, you might be tempted to change `"int"` to `"SomeType"`. This won't work, because Java has to know the actual type of the array it is trying to create. Instead use:

```
Object[] b = new Object[n];
```

This allows any type of Java object to be copied into the `b` array. Unfortunately, this will also cause an issue with this code near the bottom of the merge() method:

```
// Copy back from the temporary array
for (j = 0; j < n; j++)
  a[from + j] = b[j];
```

Because of the way the merge sort code is written, we know that all the objects in the `b` array are of the appropriate type for inserting into the `a` array … but Java doesn't know that. As far as Java is concerned, the array `b` could contain ANY type of object (because it is defined as type `Object`). To enable the code to compile, you need to change that code as follows:

```
    // Copy back from the temporary array
    for (j = 0; j < n; j++)
        a[from + j] = (SomeType)b[j];
```

Including `(SomeType)` in front of `b[j]` reassures Java that only objects of the appropriate type will be inserted into the `a` array.

The final issue is that this will generate an unnecessary compiler warning. We can suppress this warning by inserting a "`@SuppressWarnings` " clause at the beginning of the merge() method, as follows:

```
    @SuppressWarnings("unchecked")
    public void merge(...
```

Once you've made those changes, it is a good idea to add a few lines to the TimeTest class to create a ComparableDraw with, say, five tickets. Sort the array of tickets with mergeSort() and display the sorted array. Once you've confirmed that your mergeSort() is working, you can remove this simple testing code.

Update the **TimeTest** class. In the main() method, make a copy of the code you used to time the selectionSort() method with various sizes of arrays. Change this copied code to time mergeSort() for the same array sizes. The output should resemble the following:

```
*** Selection Sort ***
Quantity  Duration(ms)
========  ============
     0             0
 10000           228
 20000           704
 30000          1090
 40000          1930
 50000          3170
 60000          4384
 70000          6511
 80000          9696
 90000         12462
100000         16210

*** Merge Sort ***
Quantity  Duration(ms)
========  ============
     0             0
 10000             5
 20000             8
 30000            12
 40000            13
 50000            16
 60000            19
 70000            22
 80000            25
 90000            28
100000            34
```

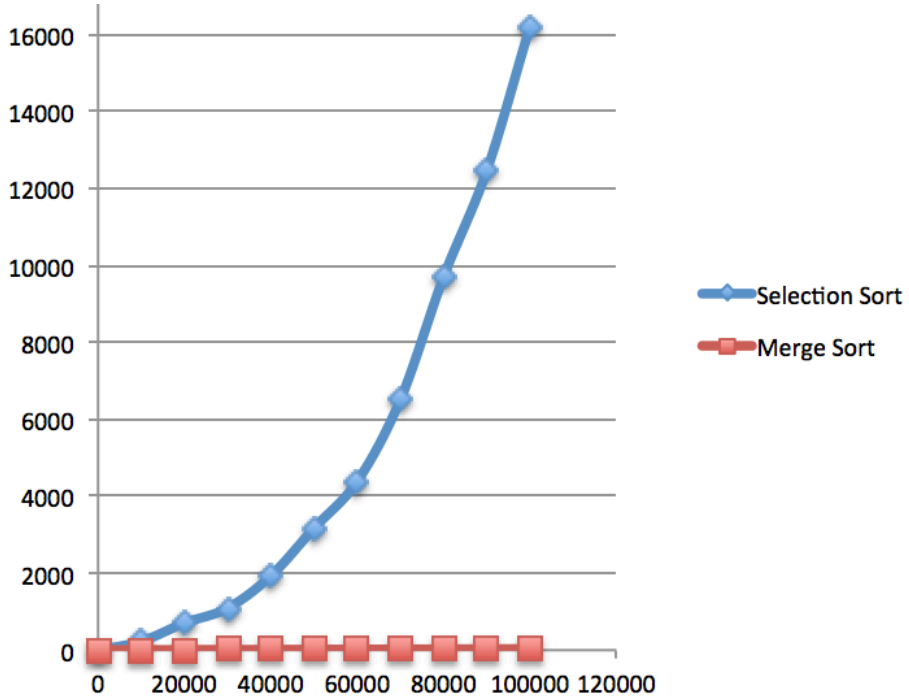
As your program executes, the hope is that you will experience first-hand how much more quickly the merge sort finishes compared with the selection sort.

Don't worry if some of the times you get for merge sort are actually smaller when N is larger – this is because the times are so small and your computer is likely also doing other work at the same time you are timing your program. This is okay; you will not lose marks because of such data values.

Using Excel, enter the values from your program's output as follows:

	A	B	C
1	Quantity	Selection Sort	Merge Sort
2	0	0	0
3	10000	228	5
4	20000	704	8
5	30000	1090	12
6	40000	1930	13
7	50000	3170	16
8	60000	4384	19
9	70000	6511	22
10	80000	9696	25
11	90000	12462	28
12	100000	16210	34
13			

Select all the data, then insert a "Smooth Marked Scatter" type of chart as you did for the Module 4 Assignment. The numbers above gave me the following chart:



You should get a graph that looks highly similar to the example above. This shows just how much faster the merge sort algorithm is compared with selection sort (or bubble sort, for that matter).

Use a screen capture to create a picture of your chart and then insert it into your submission document.

If your spreadsheet software does not have that exact type of chart, it is okay to use whichever type of chart you think is most similar to the example chart shown above.

Submission Instructions

Include the following in your submission document:

- Your name, student number, CS1083, Module 8 Assignment, and the date
- Complete code for your updated Sorter and TimeTest classes
- Sample output from executing TimeTest
- The chart you produced based on that sample output

D2L DROPBOX SUBMISSION INSTRUCTIONS

Remember to save / export your submission document as a pdf file, and then upload that one pdf file to D2L as follows:

1. In the top-navigation bar on the course screen, select 'Assessments' and then 'Assignments'.
2. Select the assignment title and follow the instructions to upload your submission document.