University of New Brunswick
CS1083 – Open Access Version

Module 4 Assignment

The purpose of this assignment is to give you practice:

- coding the selection sort algorithm;
- using the Comparable interface; and
- demonstrating the time complexity of an $O(N^2)$ algorithm.

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.

Part A: Comparable Lottery Tickets

Create a new version of the LotteryTicket class you created for the Module 2 Assignment. Call this new class **ComparableTicket**. This new class must implement the Comparable interface, which basically means you have to add a compareTo() method. Two lottery tickets are considered equal if they have the same ticket id. Otherwise the ticket with the lower ticket id value is considered to be "less than" the other ticket.

Also, modify the ComparableTicket constructor to use the IntSort class (provided earlier in this module) to sort the six numbers on each new ticket into ascending order.

Create a new version of the LotteryDraw class you created for the Module 2 Assignment. Call this new class **ComparableDraw**. Change the type of tickets held by the draw from LotteryTicket to ComparableTicket. Also add two accessor methods:

- a getTickets() method that returns a reference to the array of tickets; and
- a getTicketQuantity() method that returns the draw's ticketQty value.

(NOTE: If you were unable to successfully complete the LotteryTicket and LotteryDraw classes earlier when you worked on the Module 2 Assignment, it is necessary to complete those classes now so you can proceed with this Module 4 Assignment.)

Write a **Sorter** class containing a single method called selectionSort(). This method accepts a partially-filled array of Comparable objects and an integer that indicates how many objects are in the array. Use the selection sort algorithm discussed in the Part III instructional video to sort the objects in the array.

The video explains that selection sort can be coded two ways:

1. With each pass, find the smallest element and swap it with the element at the beginning of the unsorted portion of the array; or
2. With each pass, find the largest element and swap it with the element at the end of the unsorted portion of the array.

Use the second approach for your Sorter class, which also happens to be the approach demonstrated in the instructional video.

Use the ComparableSort class provided with this module as a template for how to code a class like Sorter.

Modify the LotteryDrawTest class you wrote for the Module 2 Assignment to produce a new version called **ComparableDrawTest**. Change all mentions of LotteryTicket to ComparableTicket, and all mentions of LotteryDraw to ComparableDraw.

Similarly to what you did for the Module 2 Assignment, create a ComparableDraw object that can hold up to 20 tickets, and loop to add 10 ComparableTicket objects to the draw. This time, however, generate random ticket id values between 1000 and 9999. This means the tickets in the draw will not initially be sorted by ticket id.

Create the same display as you did for the Module 2 assignment, showing the winning numbers, followed by a list of the unsorted tickets, the number of matches each ticket has with the winning numbers, and the dollar prize amount for which each ticket is eligible.

Use your selectionSort() method to sort the draw's array of tickets. Use the getTickets() and getTicketQuantity() methods from the ComparableDraw class to help accomplish this task.

Then repeat the same display of tickets, which should now be sorted in ascending order according to their ticket id values.

The output from executing ComparableDrawTest must be in the same format as the sample output shown below (except your randomly generated numbers will be different). Notice that the six numbers for each ticket are sorted, as are the six winning numbers.

```
Winning Numbers: 13 18 27 29 36 44

   Unsorted Tickets      #Matched    Prize
======================  ========  =========
3192:  6  8 12 15 25 46      0       $0.00
7897:  7 23 31 32 34 38      0       $0.00
4387: 14 25 26 31 35 44      1       $0.00
6556: 18 22 27 36 39 43      3     $100.00
3273: 15 16 24 36 38 39      1       $0.00
8349: 13 31 32 34 37 40      1       $0.00
8112:  6 11 17 22 23 26      0       $0.00
4838:  8 13 29 36 46 49      3     $100.00
4600:  9 10 22 35 48 49      0       $0.00
7806:  5  6  7  8 39 42      0       $0.00

    Sorted Tickets        #Matched    Prize
======================  ========  =========
3192:  6  8 12 15 25 46      0       $0.00
3273: 15 16 24 36 38 39      1       $0.00
4387: 14 25 26 31 35 44      1       $0.00
4600:  9 10 22 35 48 49      0       $0.00
4838:  8 13 29 36 46 49      3     $100.00
6556: 18 22 27 36 39 43      3     $100.00
7806:  5  6  7  8 39 42      0       $0.00
7897:  7 23 31 32 34 38      0       $0.00
8112:  6 11 17 22 23 26      0       $0.00
8349: 13 31 32 34 37 40      1       $0.00
```

Part B: An Experiment

Create a new class called **TimeTest** with a main() method that performs the following testing.

Use a for loop to test each quantity starting with 0 and incrementing by 10,000 up to 100,000.

For each quantity value:

- Construct a new ComparableDraw instance that can hold that quantity of ComparableTicket objects;
- Fill the draw with that quantity of tickets. Randomly generate the ticket id values as described in Part A of this assignment, which means those values will be unsorted;
- Use your selectionSort() method to sort the tickets;
- Immediately before and after sorting, use System.currentTimeMillis() to capture the current system time on your computer. Subtract the "before" time from the "after" time to get the duration that it took to perform the sort operation (a number of milliseconds); and
- Display the quantity and duration.

The output from executing your TimeTest class should look similar to the following example:

```
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
```

To align the columns as shown, try using the Java printf command, similar to the following line from my sample solution for this assignment:

```
System.out.printf("%6d%12d\n", qty, duration);
```

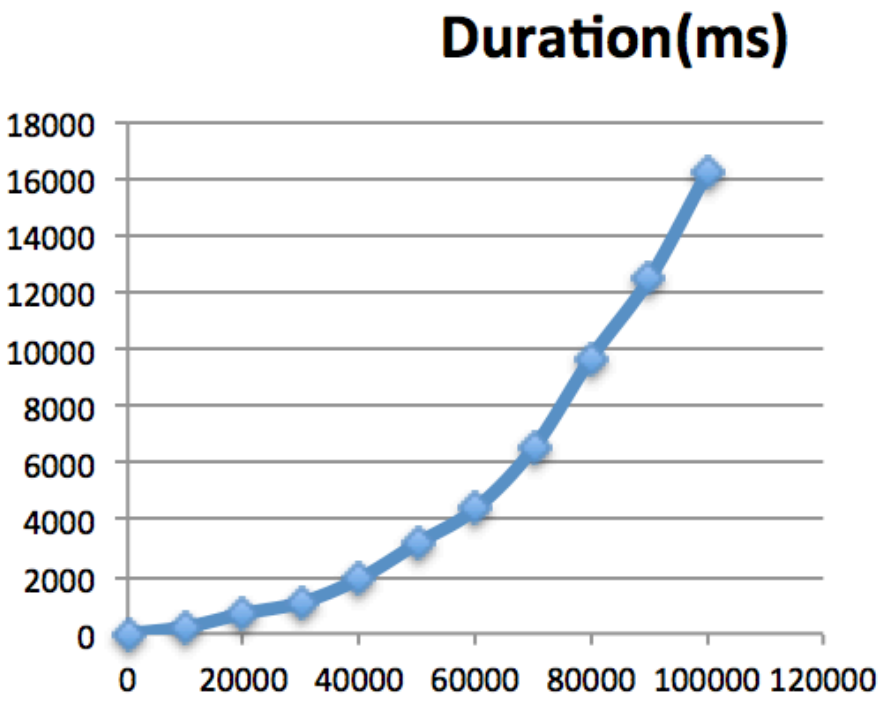
A String like `"%6d%12d\n"` is how we can instruct Java on how to display the two values that follow (`qty` and `duration`).

`%6d` directs Java to display the numeric value of `qty` right-justified with a width of 6 characters. The `d` stands for the somewhat old-fashioned term "decimal".

Similarly, `%12d` directs Java to display the numeric value of `duration` right-justified with a width of 12 characters.

`"\n"` is, of course, Java's newline character.

As the last step for Part B, enter the values from your program's output into a spreadsheet and generate a chart. Here is an example chart I produced from the sample output shown above.



The hope is that you will see the pattern of exponential time growth we expect with an $O(N^2)$ algorithm like selection sort. The example above exhibits this pattern. You might get some data values that don't fit this pattern exactly – this is because 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 if your data values produce a less-than-perfect exponential graph.

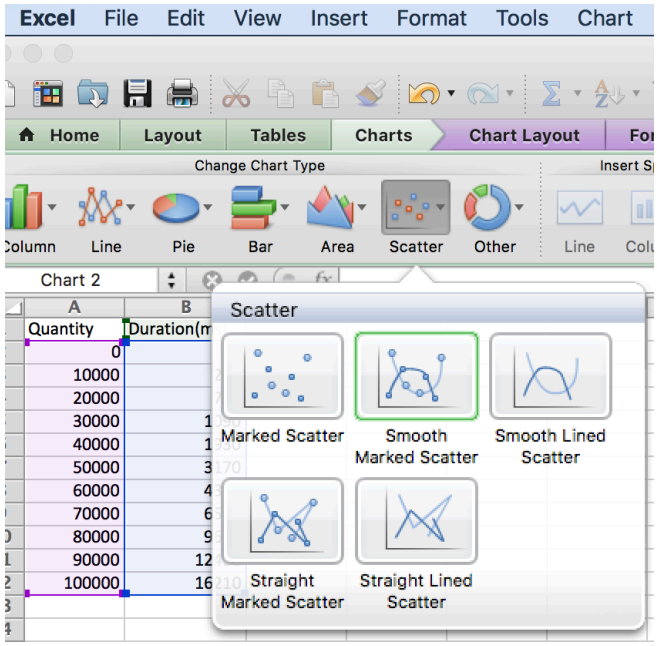
If you don't have a spreadsheet application, UNB students can access Microsoft 365 for free, which includes Microsoft Excel. Login to my.unb.ca and use the Microsoft 365 link on the Launch Menu. OpenOffice is also free for everyone.

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

	A	B	C
1	Quantity	Duration(ms)	
2	0	0	
3	10000	228	
4	20000	704	
5	30000	1090	
6	40000	1930	
7	50000	3170	
8	60000	4384	
9	70000	6511	
10	80000	9696	
11	90000	12462	
12	100000	16210	
13			

Select all the data (as shown above), then Insert, Chart…

I selected the "Smooth Marked Scatter" type of chart (shown below), which produced the example chart shown above.



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 4 Assignment, and the date
- Part A:
  - Complete code for your ComparableTicket, ComparableDraw, and ComparableDrawTest classes
  - Sample output from executing ComparableDrawTest
- Part B:
  - Complete code for your TimeTest class
  - 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.