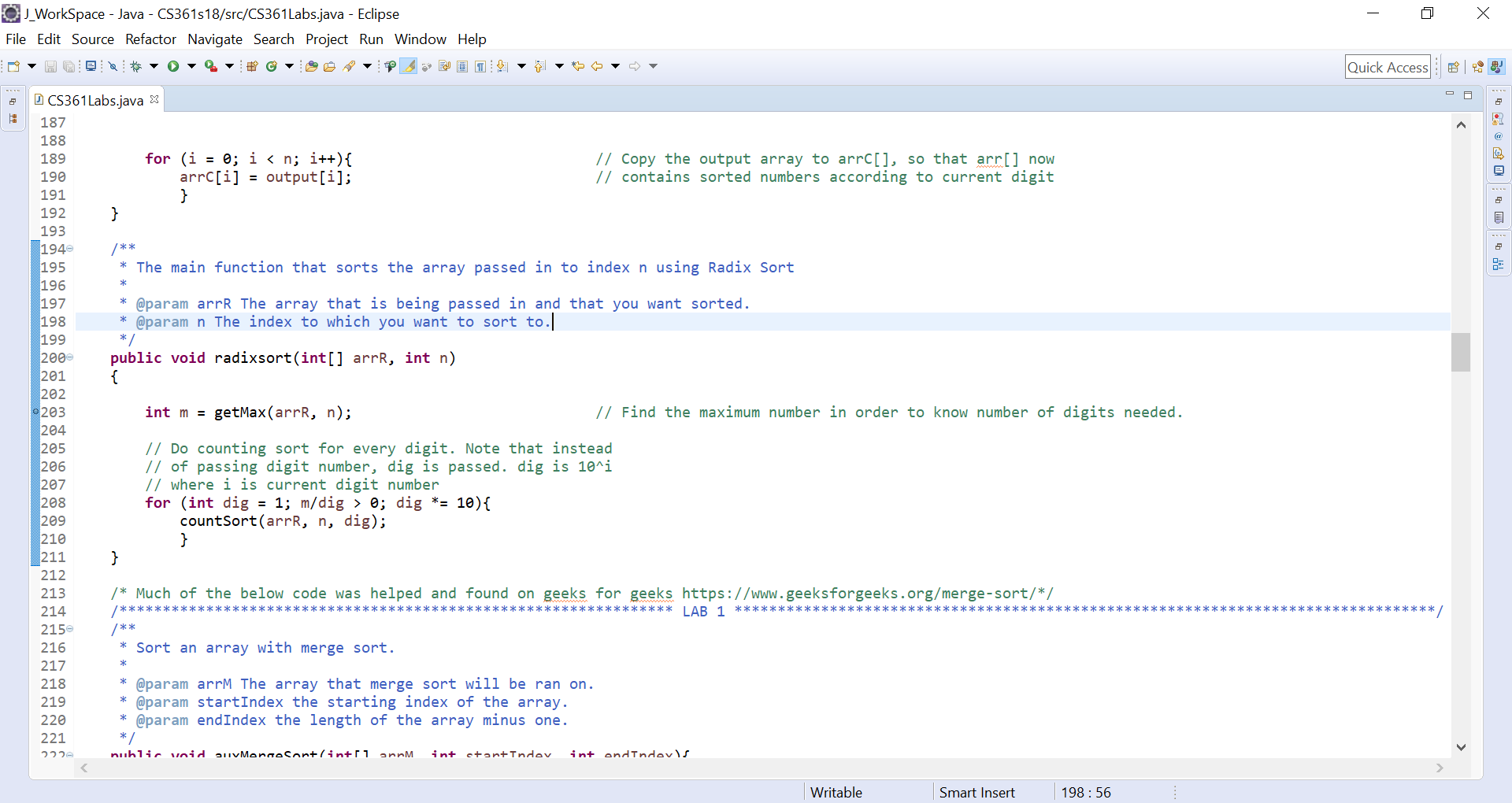
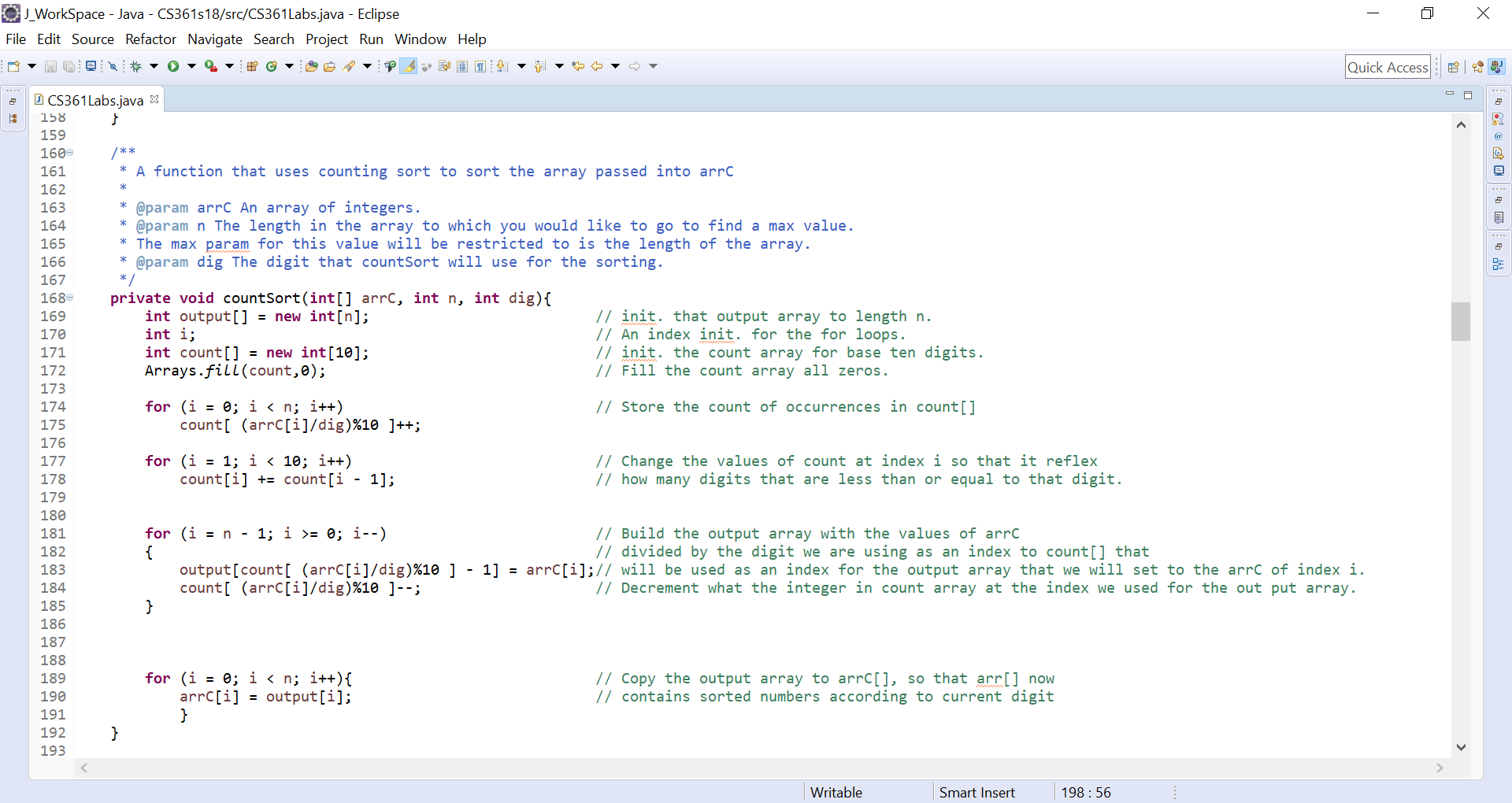
# CS361 Algorithm Lab 2

# What to do

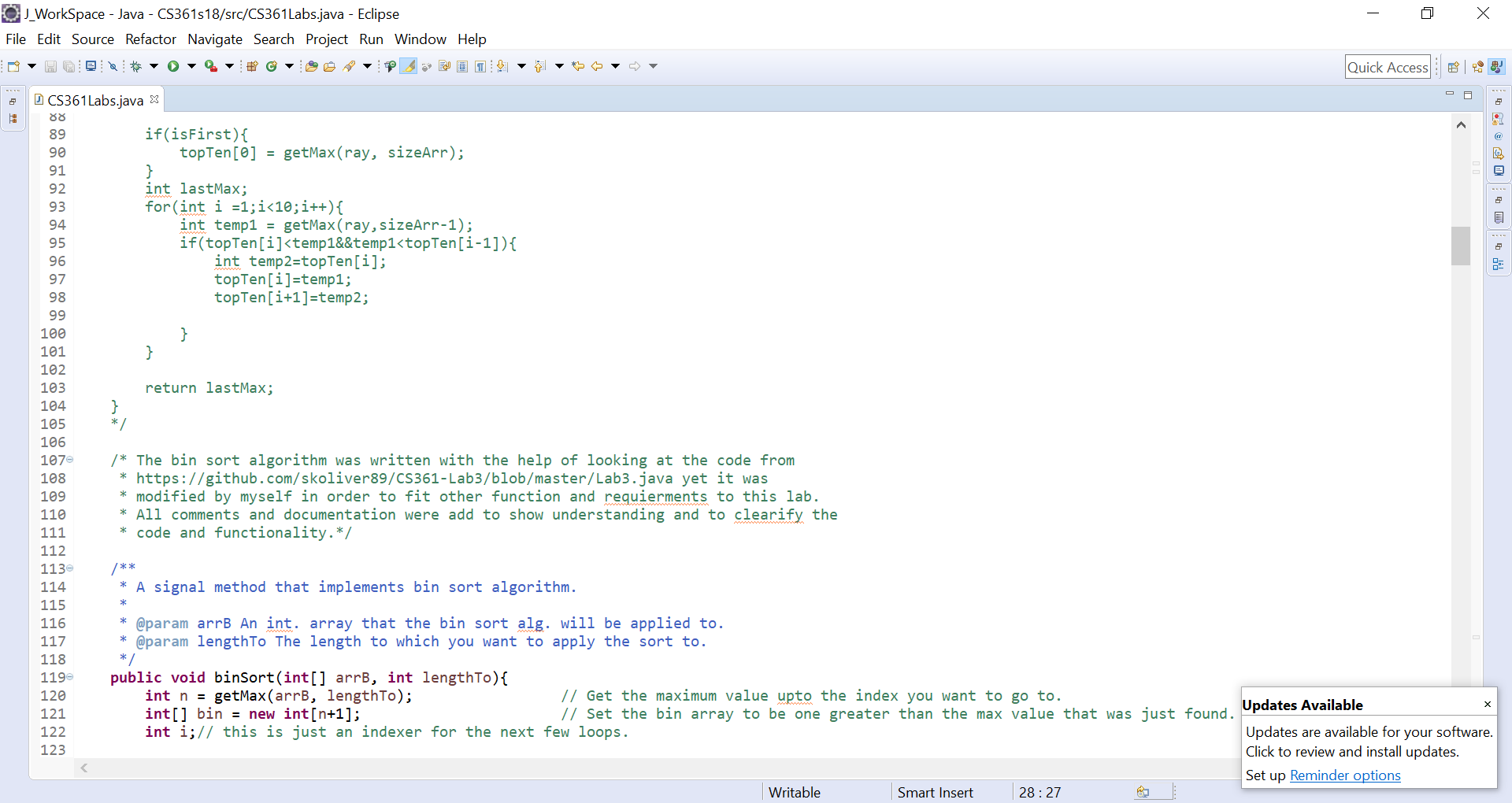
1. Implement the Radix sort algorithm and use it to sort roughly 10,000,000 numbers. I am providing a new data file (on Moodle).

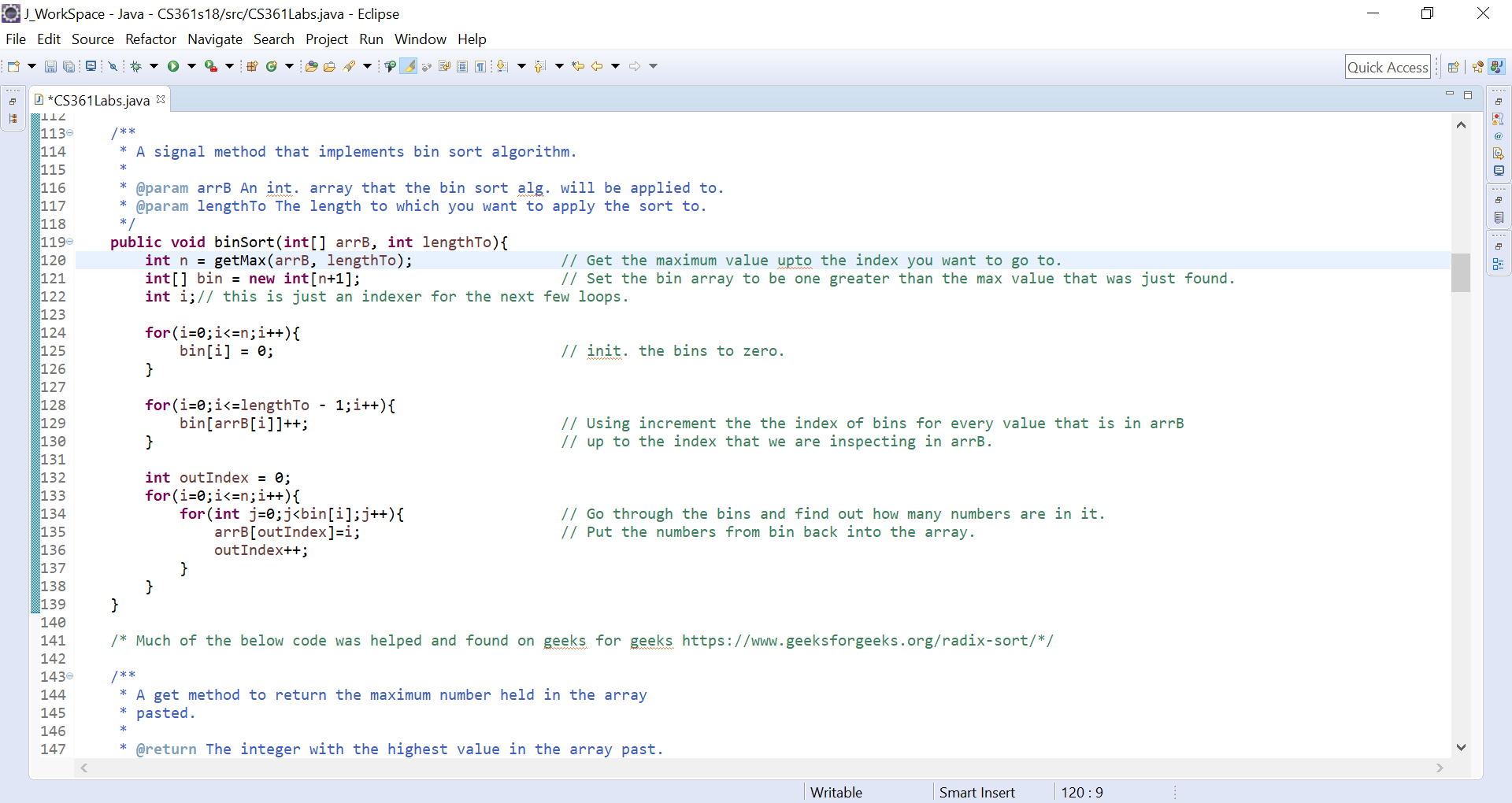


**Below is the counting sort alg. That was used for my radix sort.**

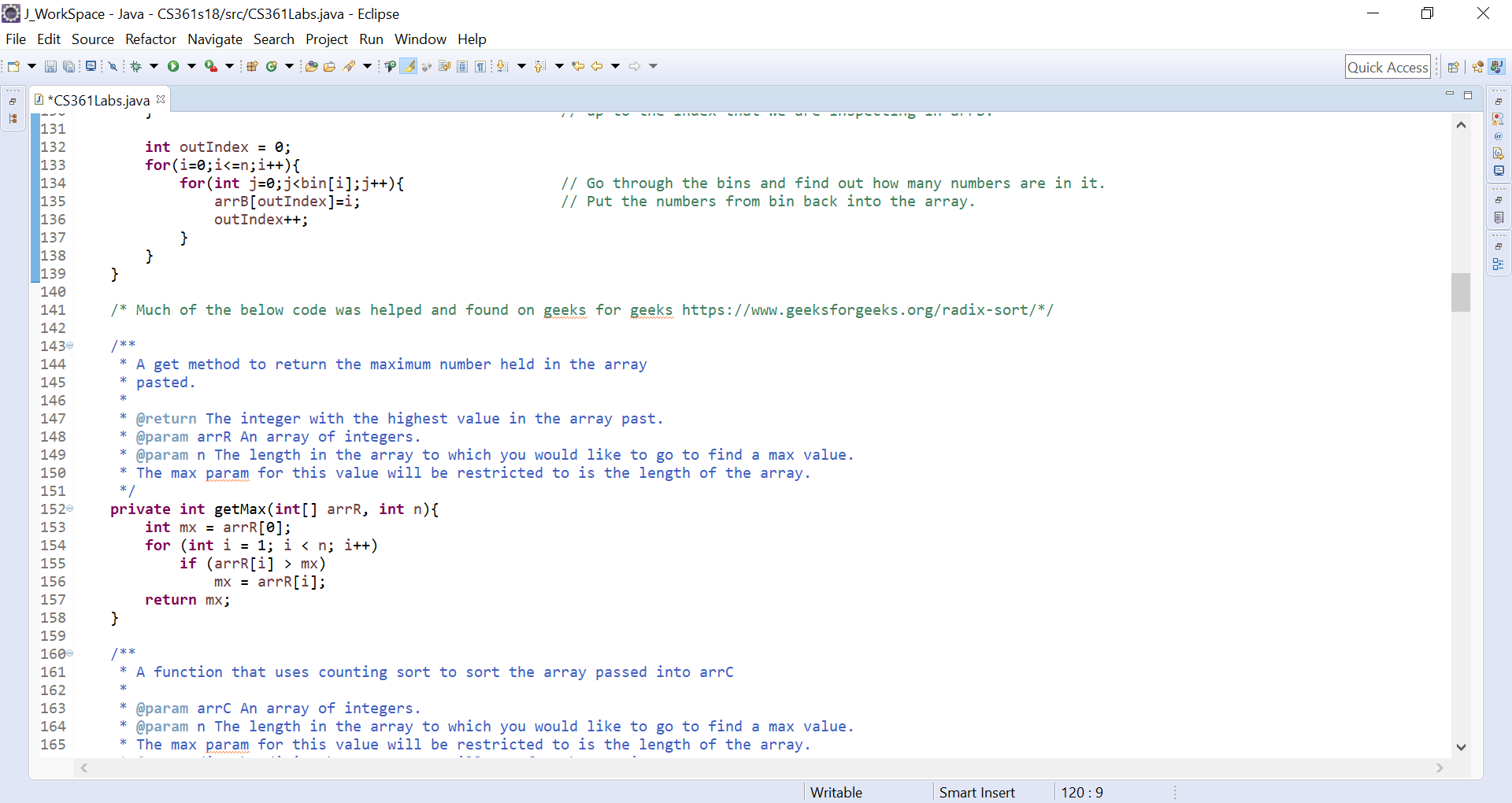


1. Implement the Bin sort algorithm and use it to sort roughly 10,000,000 numbers.

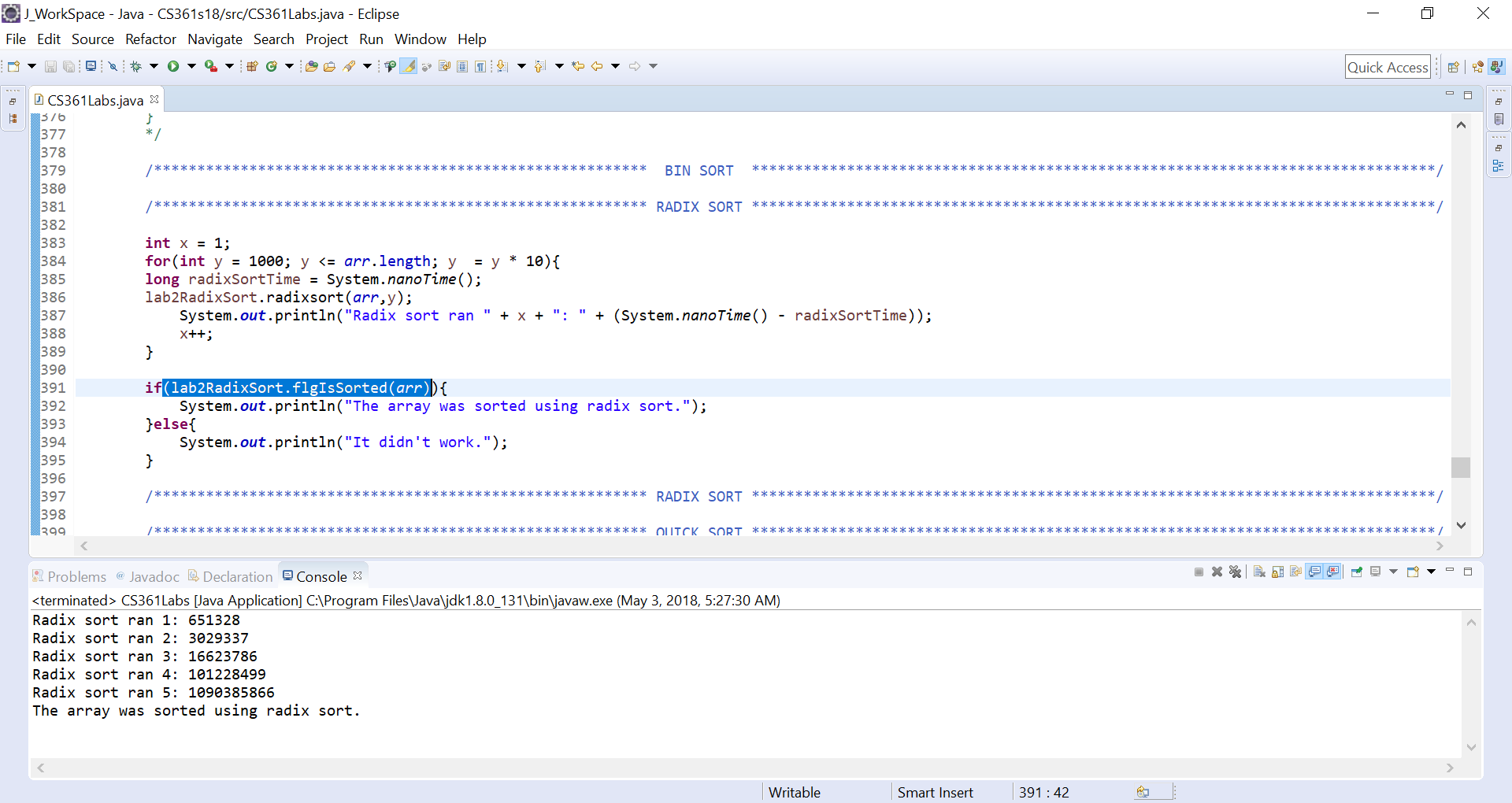




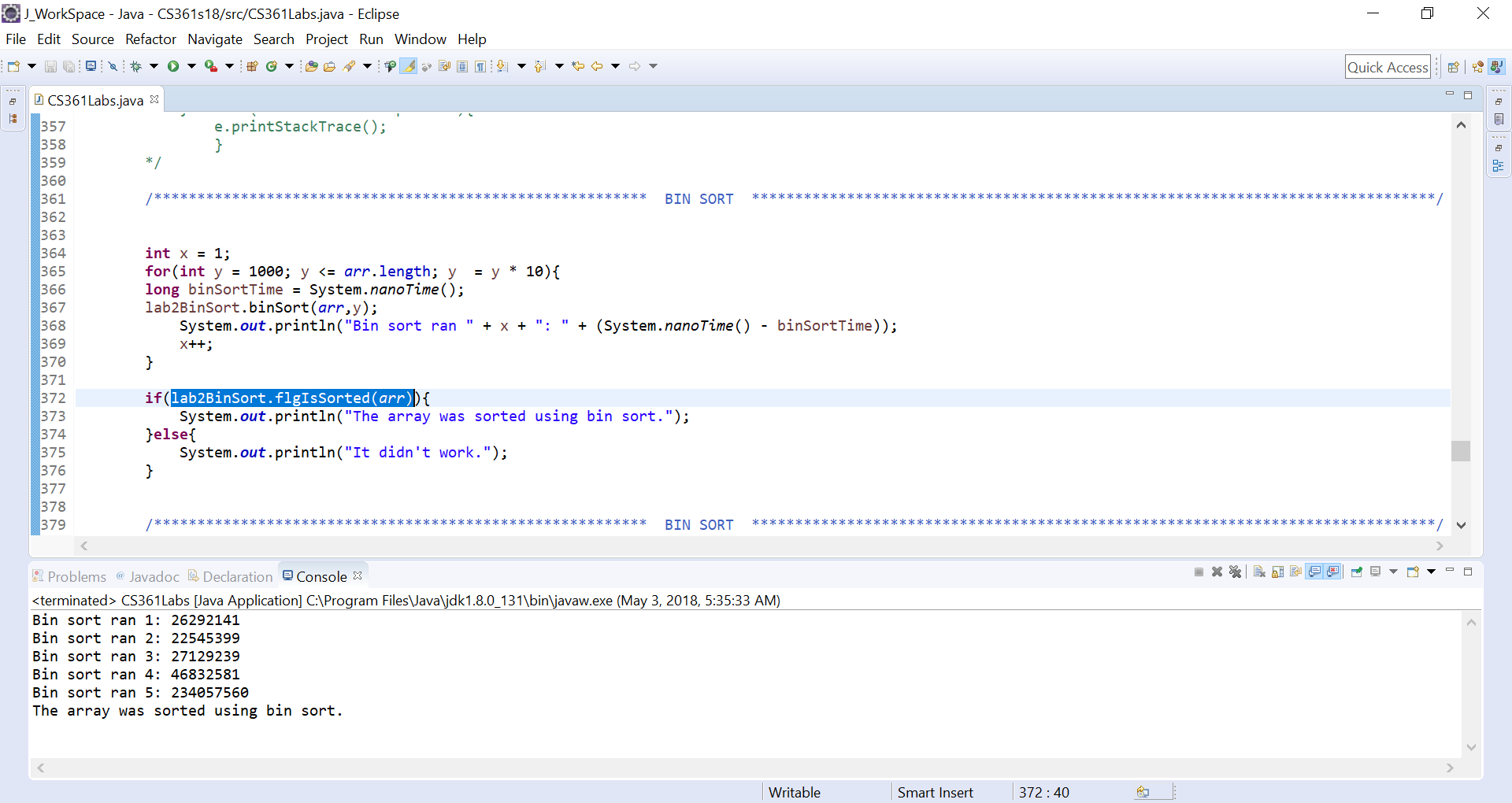
**Below is the method that allows me to retrieve the max value and us it in my bin sort.**



1. Make sure the results are sorted for 1 and 2. Show the screen dump indicate the sorting algorithms are actually sorting correctly.

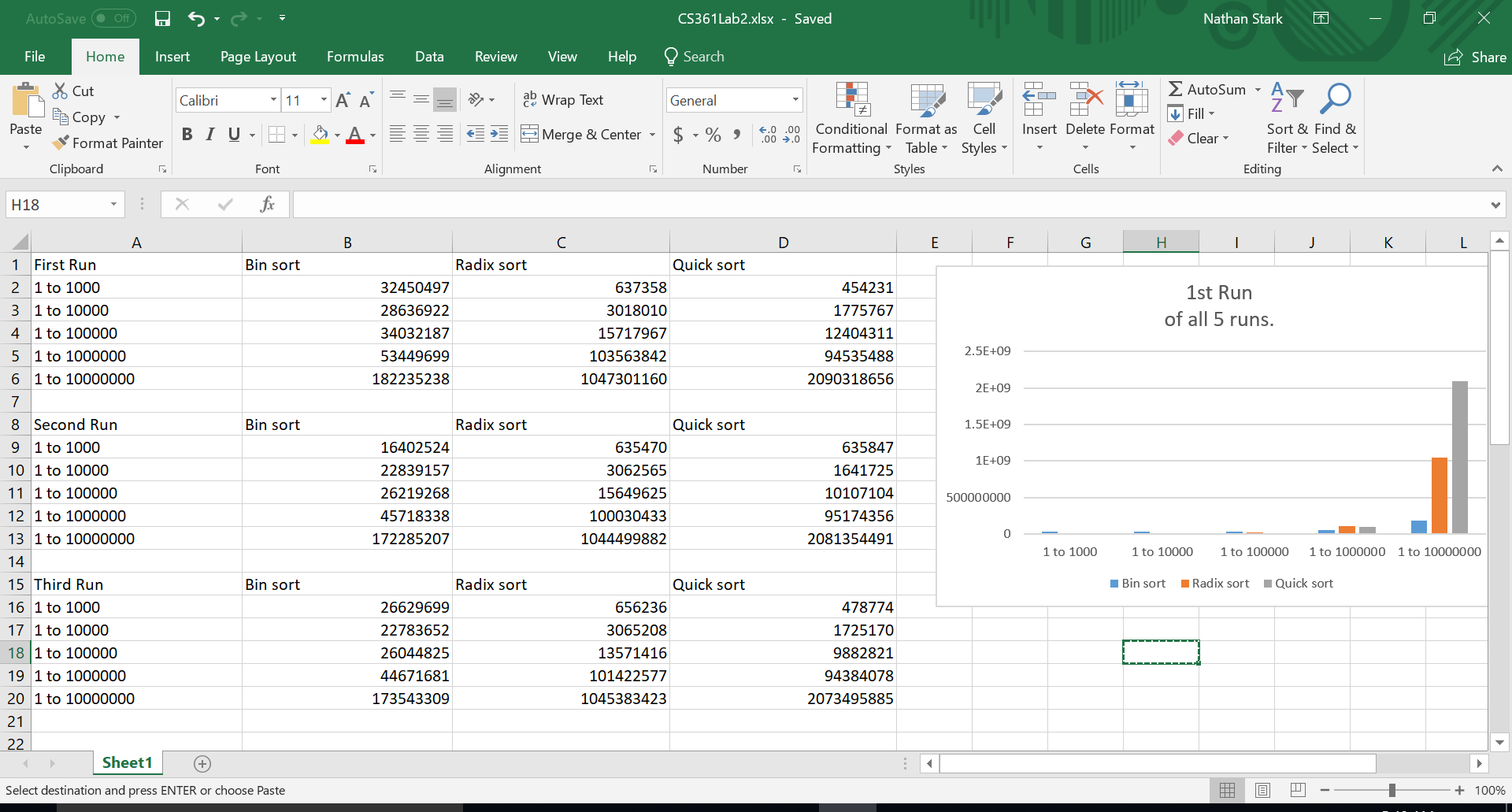


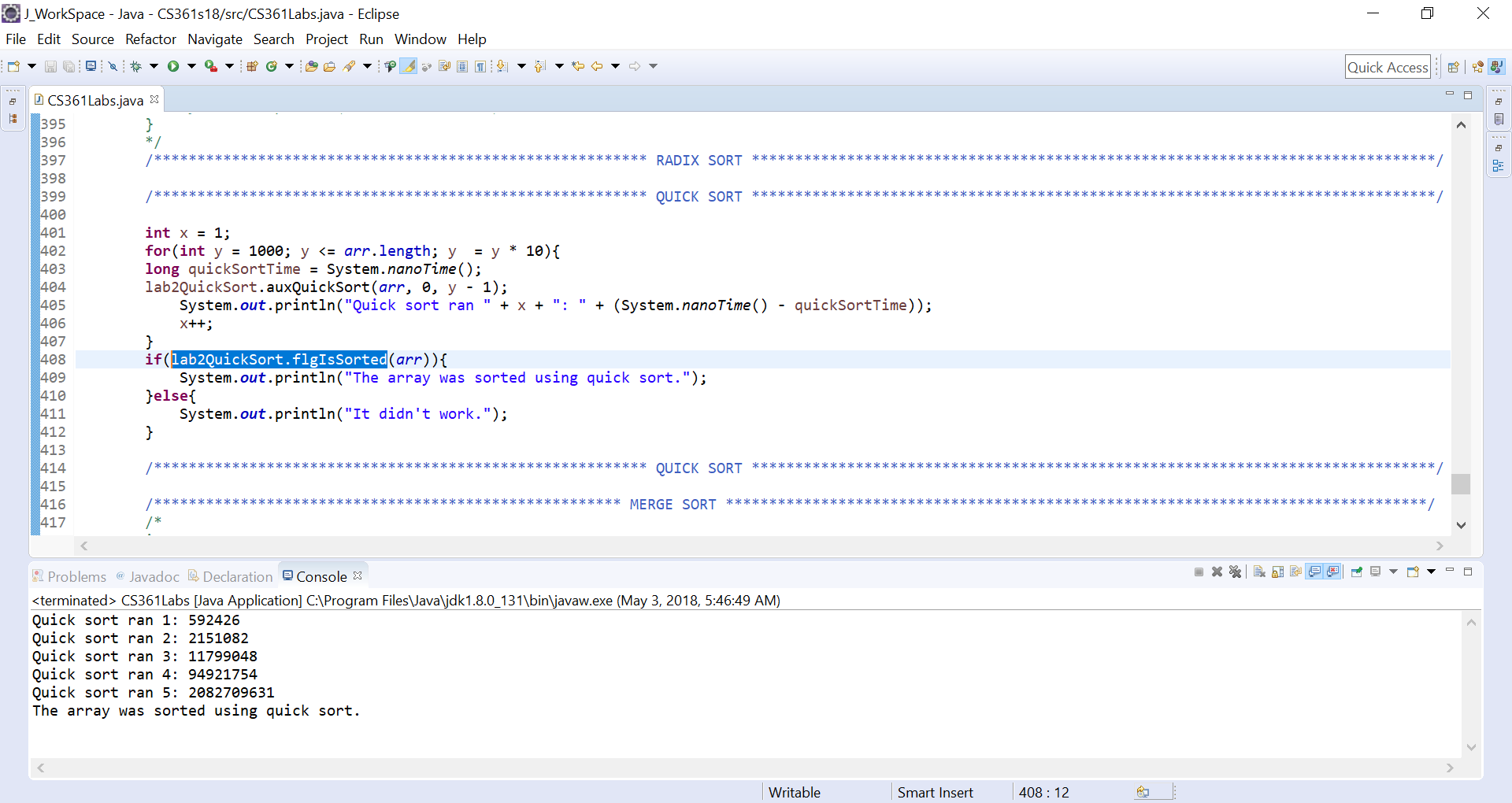
**As we can see the flgIsSorted() method is use in an if statement so that the statement “The array was sorted using radix sort.” Will print to the console only if the array is sorted. Otherwise the statement “It didn’t work.” Prints to the console. As we can see the proper statement is printed.**



**As we can see the flgIsSorted() method is use in an if statement so that the statement “The array was sorted using bin sort.” Will print to the console only if the array is sorted. Otherwise the statement “It didn’t work.” Prints to the console. As we can see the proper statement is printed.**

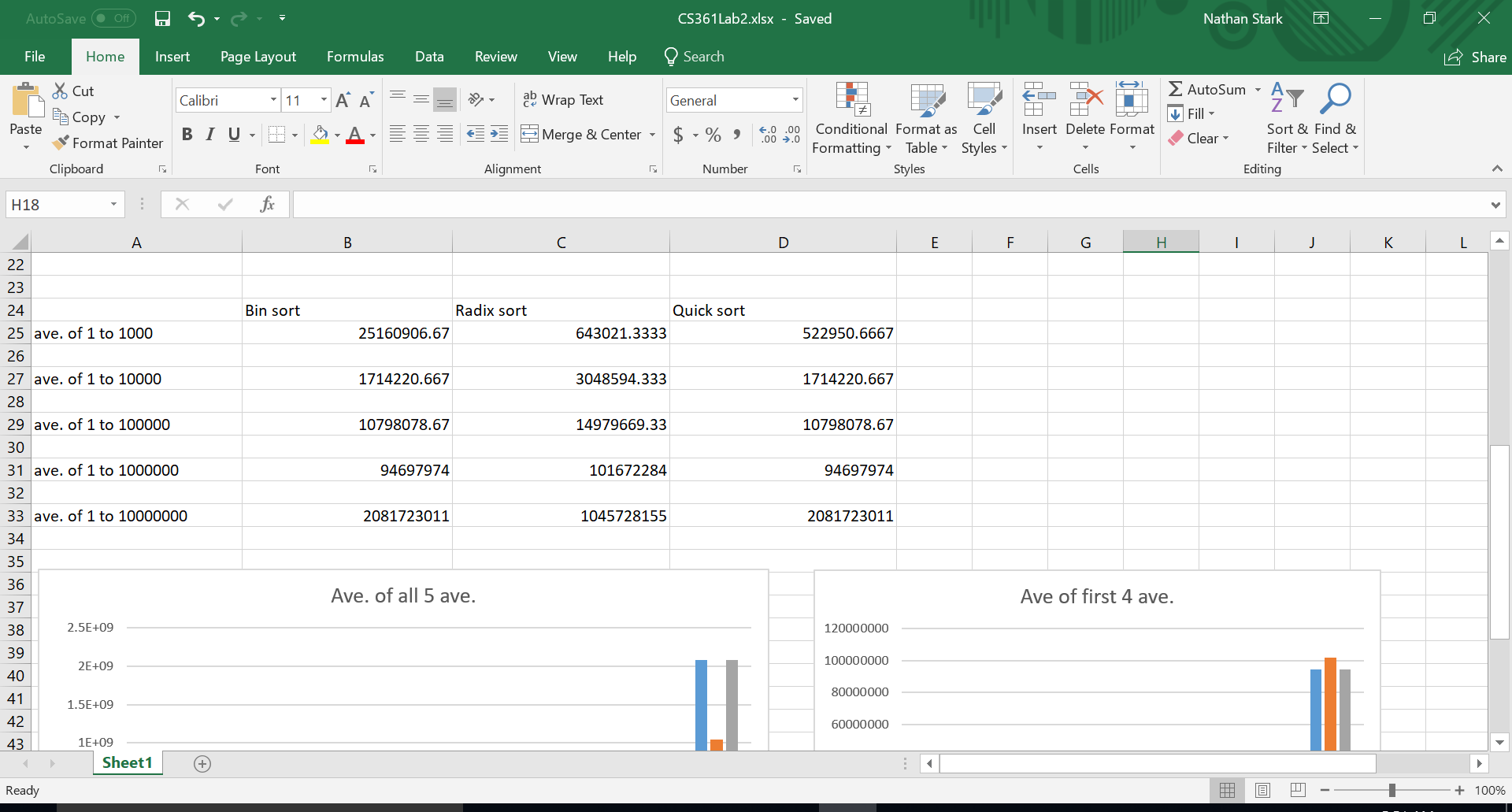
1. Show the execution time comparison with your either quick sort or merge sort. Also make sure the result of your quick sort or merge sort is sorted.



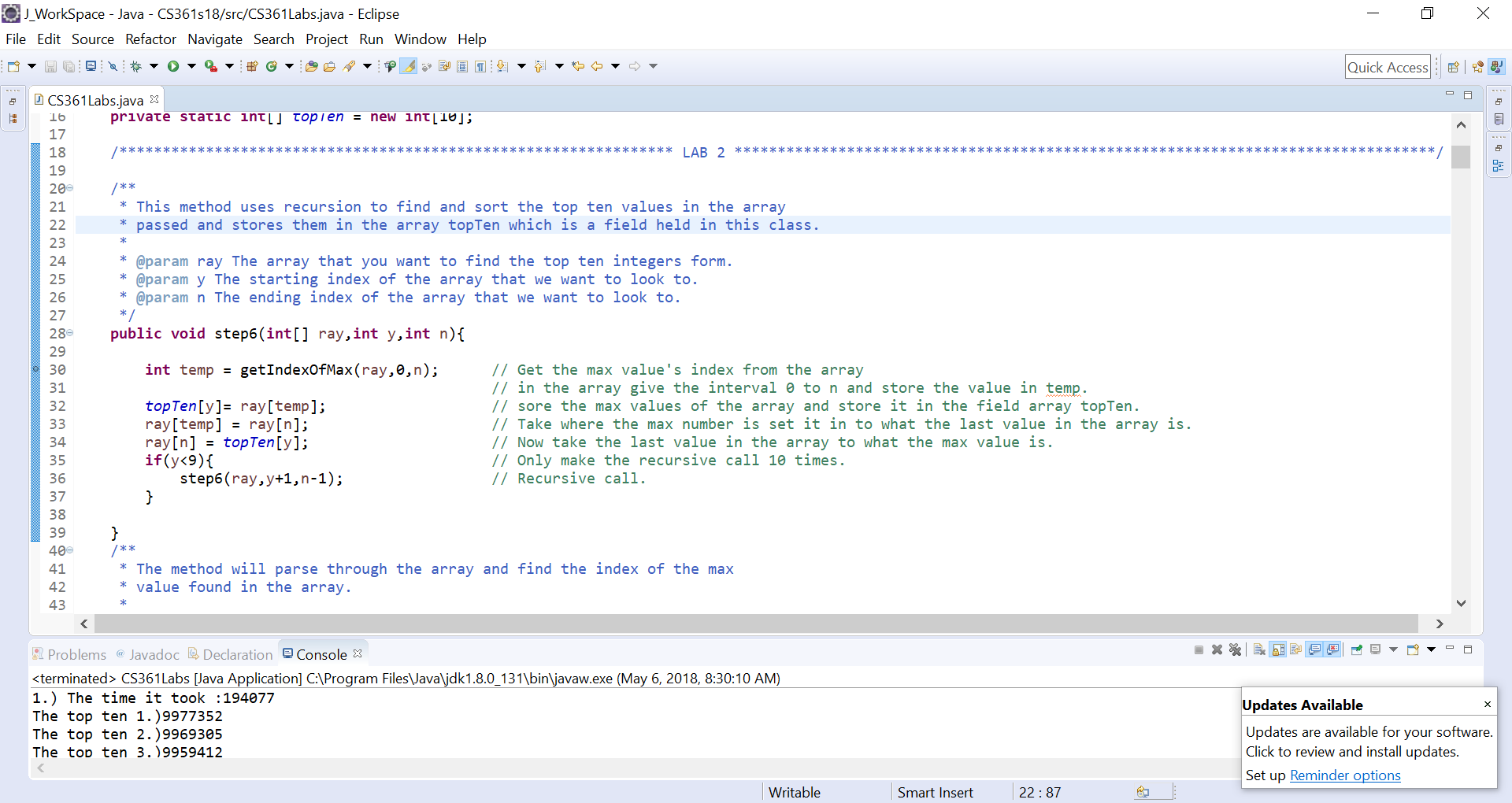


**As we can see the flgIsSorted() method is use in an if statement so that the statement “The array was sorted using quick sort.” Will print to the console only if the array is sorted. Otherwise the statement “It didn’t work.” Prints to the console. As we can see the proper statement is printed.**

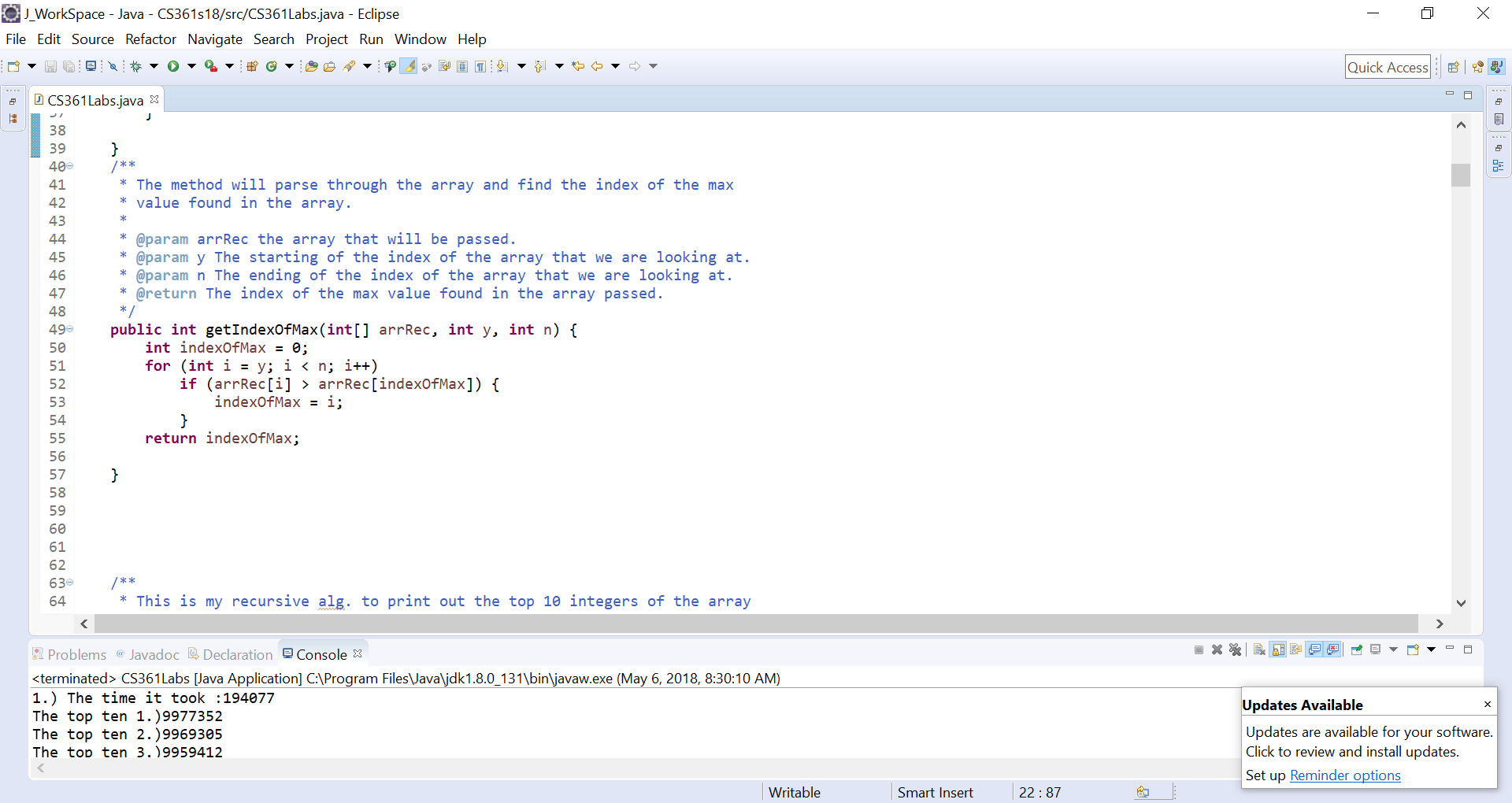
1. Run your code for 1~3 three times, record the execution time in milliseconds for each run on each size, enter the milliseconds reading into an Excel spreadsheet, calculate the average execution time in milliseconds and provide your results in a table and/or as a line chart.

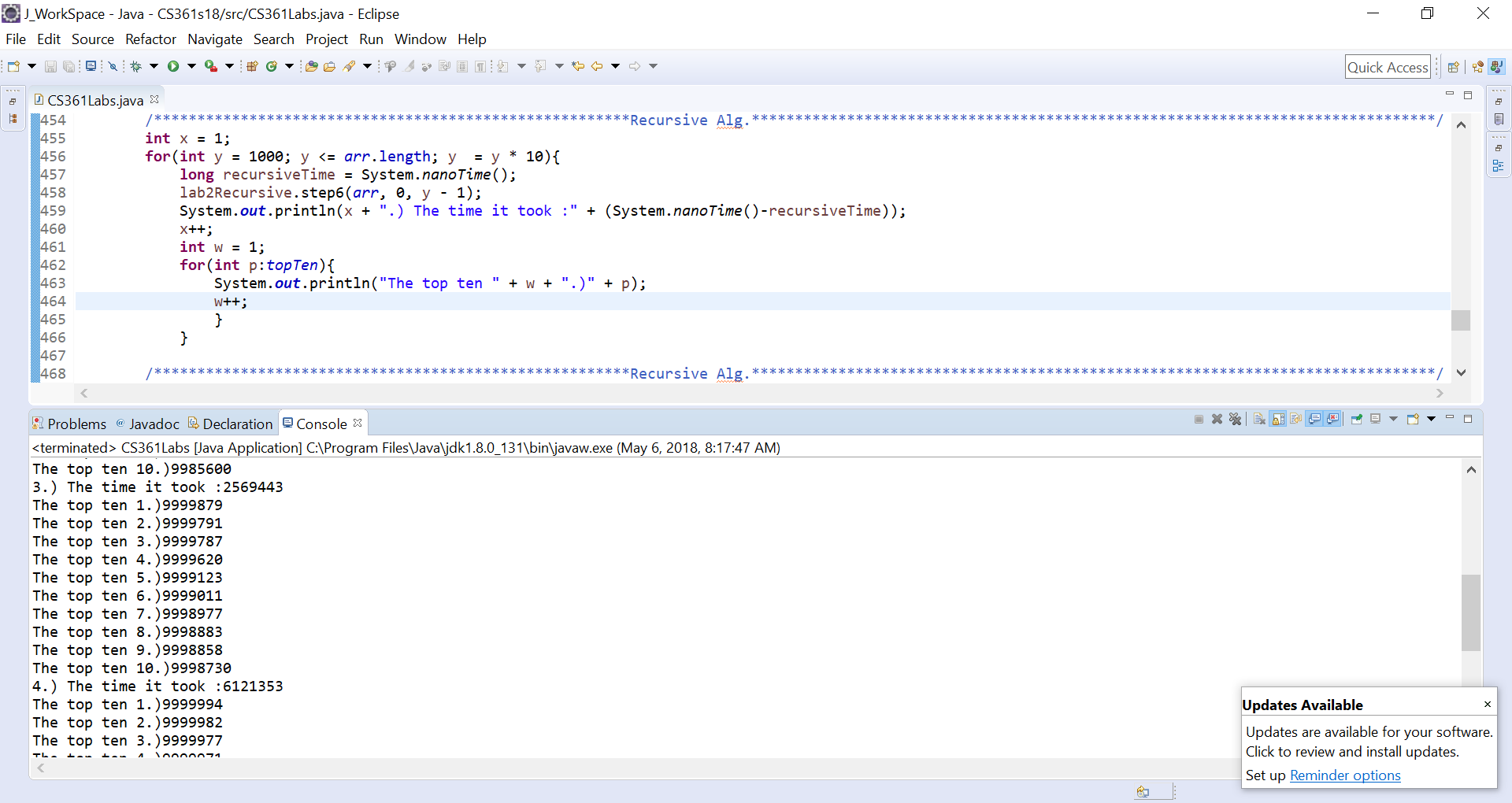


1. Use your Lab 1 read method to from my data file. Then write **recursive** algorithm to list the largest 10 elements of the data you read, and listing them in decreasing order as the output. Again, start with 1,000 and increases at 10x until it needs to read more than 10 million numbers. Output the execution time of your approach.

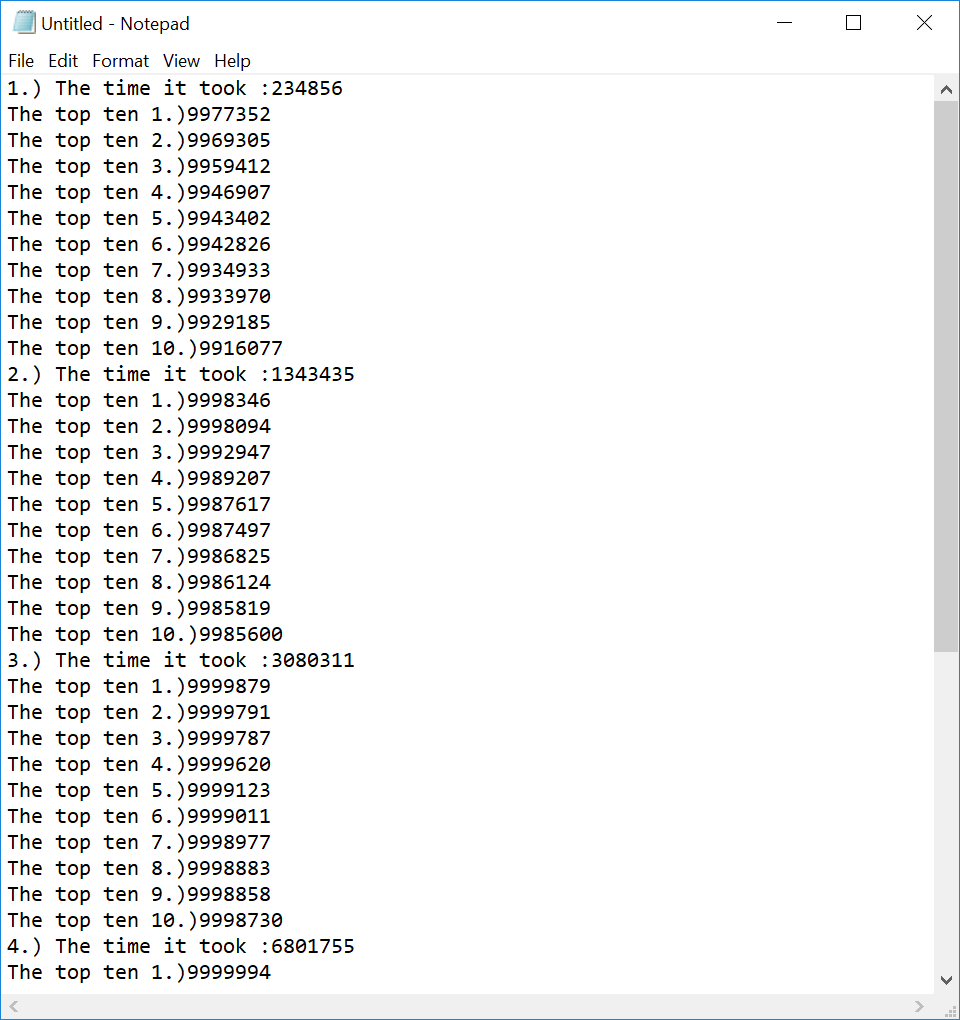
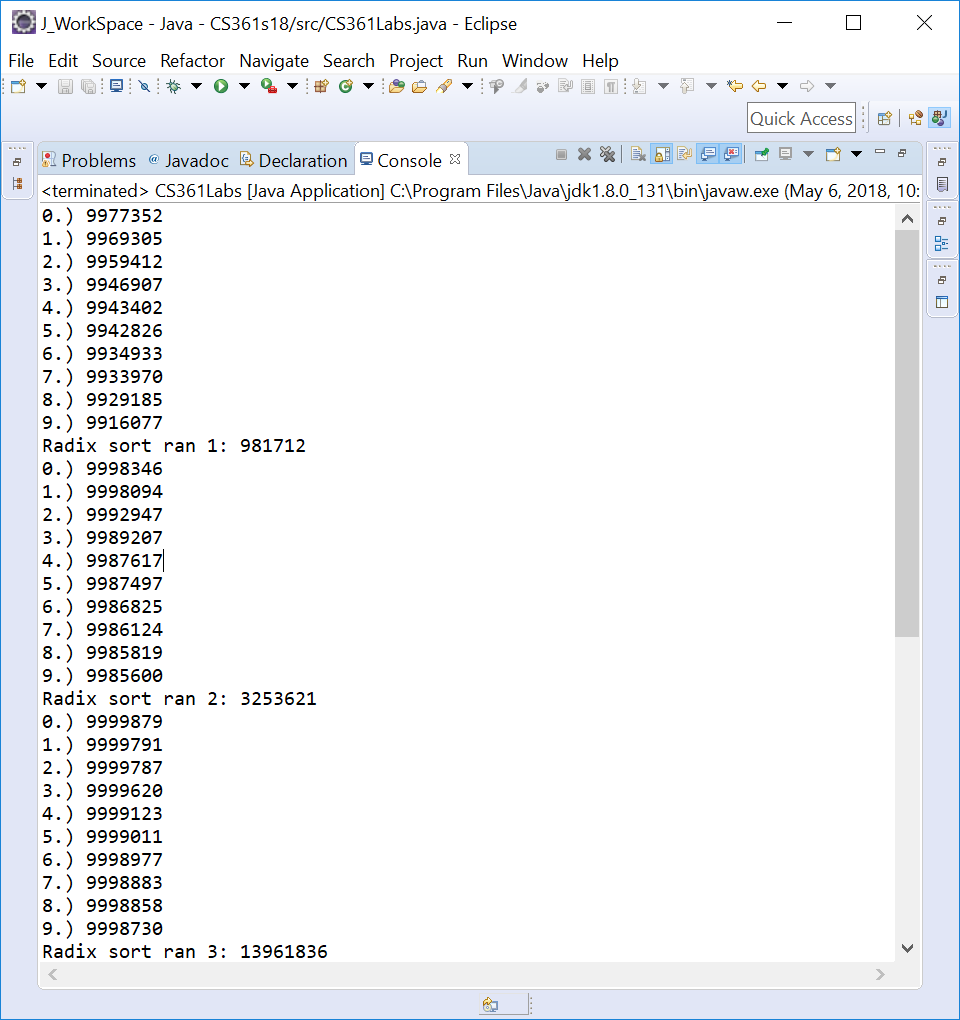


**Above is the recursive method that I used to sort the top ten integers in descending order. Below is a helper method for the recursive method.**





1. Test your result by calling one of your sorting algorithm to sort the data first and display largest numbers in decreasing order as the output. Output the execution time of your approach.

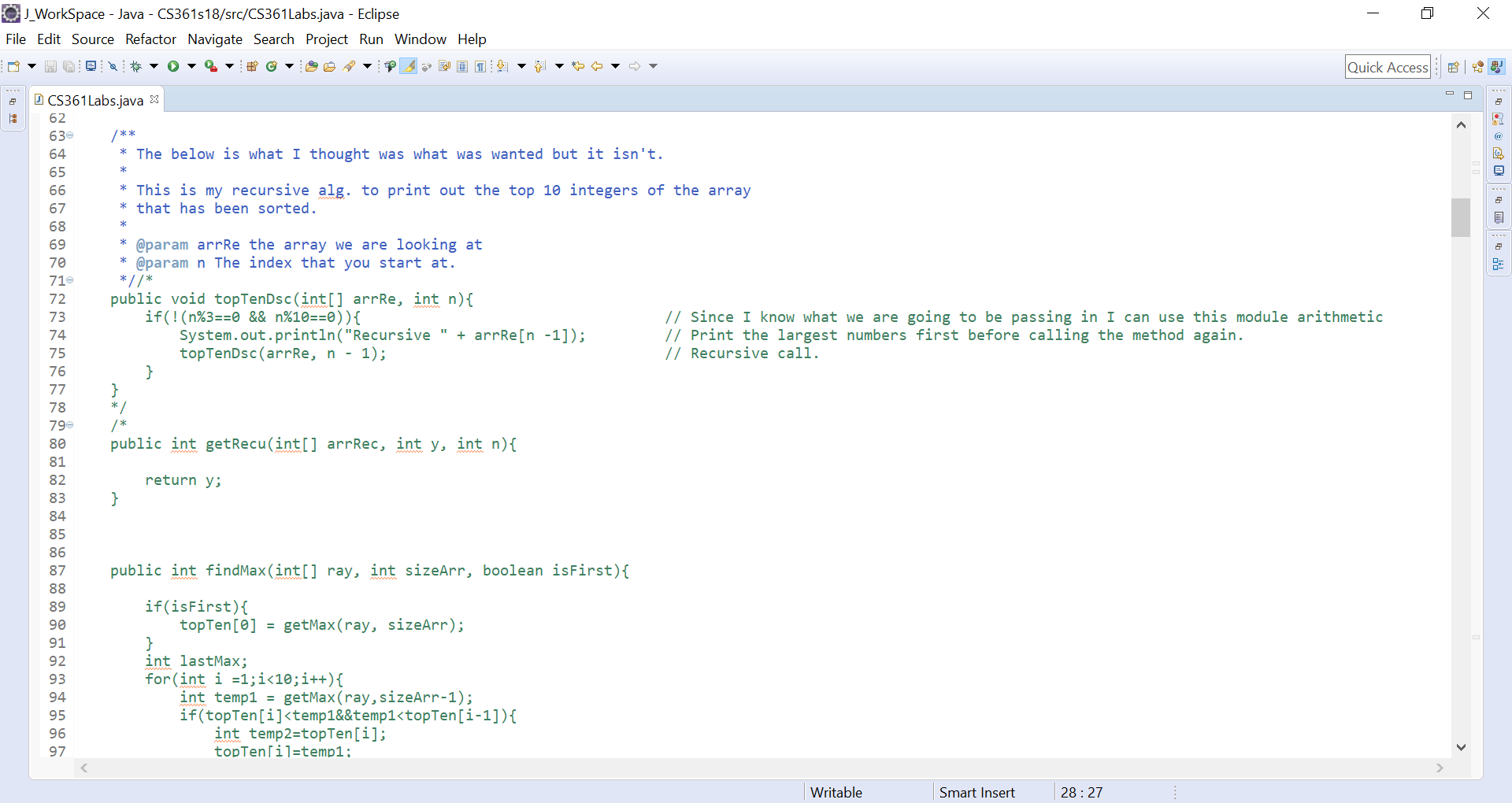


**On the right is the print out of the recursive method that I wrote and executed to print out the top ten values in decreasing order. On the left is the same thing only radix sort is used to sort all of the integers in the array before printing the top ten values in decreasing order.**

1. Run your code for part 6 and 7 three times, record the execution time in milliseconds for each run on each size, enter the milliseconds reading into an Excel spreadsheet, calculate the average execution time in milliseconds for each run on each size and display your results in both a table and as a line chart.

**The above is them side by side. In order to have a better perspective I will show them independently.**

1. Write a half to one-page report to explain your execution time observation and discuss the problem-solving approach you applied for step 6. Is it DP, greedy algorithm, or divide-and-conquer?

**To solve the problem, I went through many executions and approaches that ended up with stack over flow errors, or I was solving the wrong problem as seen below in the commented-out code.** **Through all of them I was trying dynamic programming to store the values into an array (topTen) of size 10. I was then reminded of the pivot strategy that we used for partition method, so I thought that I could use that kind of methodology to solve this problem. Only worrying about the ten values that I care or have been tasked to find. I also employed divide-and-conquer as I separated the max value out of the array that I was parsing each time. This made my problem smaller each time, and focused on the new problem, finding the next maximum value.**

**My execution time for my helper method to return the index of the max is O(n) then I would multiply that by 10 because we call that method 10 times in the step 6 main method + some of the steps that are taken in the execution of both the methods, which we will ignore as this is in respect to time complexity. So, the time complexity is O(n) time.**