CS 122 – Lab #14 – Sorting algorithms

In today's lab, you will complete the implementations of some sorting algorithms in Java. Then, you will test these algorithms, and compare their relative efficiency. Are some ways to sort faster than others?

Create a new directory called lab14 in your account. Download the files Array.java, Driver.java, Generate.java, and Tester.java to this directory. Remember that you can copy these files directory: their location on the server is ~chealy/www/cs122/lab14.

Part 1 – Implementation

Inside Array.java I have provided the implementation of two sorting methods: bubble sort and swap (i.e. exchange) sort. You need to complete the implementations of two other sorting methods: selection sort and insertion sort. There is a comment above each empty method that gives you a hint about how each sorting method should work.

When you are ready to test your implementations, run the Tester program. Enter the command "java Tester". The Tester class has an ad hoc small array that runs through each sorting algorithm. You should observe that in each case, the array is correctly sorted.

Have your algorithms checked by the instructor or lab aide before going on to the next section. √

Part 2 – Execution Time

Now that your implementations are correct, let's determine their execution times. We need to run each sorting algorithm with larger input. The main program in Driver.java is similar to Tester.java. The main difference is that we do not print the contents of the arrays after they are sorted. Here, we are interested in their execution times, which is measured in milliseconds.

How do we get a large input array? This is where Generate.java comes in. This program will automatically generate a randomly scrambled list of integers that can serve as suitable input for Driver.java. Generate.java expects one command-line argument: a positive integer, n. It will then create a list of numbers from 1 to n, shuffled. This output needs to become the input to Driver.java. Therefore, you need to run the program this way in order to see the results:

```
java Generate 100 | java Driver
```

This will run your sorting algorithms on arrays of size 100. Fill in the following table with your results. What happens to the execution time of each sorting algorithm if we double the size of the array?

Array size	Selection time (ms)	Insertion time (ms)	Bubble time (ms)	Swap time (ms)
5000				
10,000				
20,000				
50,000				