Assignment A9 (30 marks)

Focus: *Sorting*

In this assignment, you will build a class that can be used to sort a list of patient using different algorithms, and you will also compare the time efficiency of these algorithms.

Download the attached files to help you work on this assignment:

- `Patient.java`
- `PatientComparator.java`
- `PatientTestQ12.java`

Q1. **[10 marks]** Create a class called `Sorter` that has two static methods:

public static void bubbleSort(ArrayList<Patient> list)
public static void bubbleSort(ArrayList<Patient> list, Comparator<Patient> comparator)

Write code for both methods so that they can sort an list of items of the type `Patient` using bubble sort. The first one should use the `Comparable` interface and the second uses the `Comparator` interface.

Test your code using the attached file `PatientTestQ12.java`. You should get an output similar to the one given below for either method.

Sample run (the asterisk indicates a patient with an emergency)

```
Before sorting    [p1, p2, *p3, p4, *p5]
After sorting     [*p3, *p5, p1, p2, p4]
```

Q2. **[10 marks]** Add the following two methods to your `Sorter` class. The first method uses the selection sort algorithm and the second uses insertion sort. Both methods should use the `Comparable` interface.

public static void selectionSort(ArrayList<Patient> list)
public static void insertionSort(ArrayList<Patient> list)

Test your code again using `PatientTestQ12.java`. You should have the same output as in Q1 above.

Q3. **[10 marks]** Write a program that obtains the execution time of the three sort algorithms used in the above `Sorter` class (i.e., bubble sort, selection sort, and insertion sort). Your program should print out the time required to sort array lists of N patients, where N ranges from 5,000 to 50,000 with an increment of 5,000 (see sample run below). Every time increment N, your program should recreate unsorted array lists of N random patients and then sort them. A random patient should have a random id between 0 and N and random

emergency case (`true` or `false` for `emergencyCase`). Don't worry much about creating a random name for each patient. Instead, use the name "anonymous" (or any other name of your choice) for all patients.

In order to properly compare the performance of the three sorting algorithms in `Sorter`, you need to have them work on three identical array lists. Start by creating an array list with random patients, then clone twice. Finally, sort each array list (original and clones) with a different algorithm and print the sorting time. Repeat this process for different values of N.

When you submit your code, you need submit two screen shots of the output of two runs of your code (similar to the ones given below).

Hints:

- To add randomness to your program, you may use the methods from the `Random` class or the `Math.random()` method. `Random` class has methods to generate random values of many primitive types (`int`, `double`, `boolean`, etc). However, `Math.random()` only generates random double numbers greater than or equal to 0.0 and less than 1.0. If you decide to use `Math.random()`, you need to think about how to generate random integers for `id` and random `boolean` values for `emergency`.
- To create three identical lists, you can use the `clone` method.
- To better organize your code, create a helper method called `randomPatient` that returns a new instance of `Patient` with random attribute values.
- To measure the execution time of a *task*, you can use the following template

```
long startTime = System.currentTimeMillis();
perform the task;
long endTime = System.currentTimeMillis();
print (endTime - startTime);
```

- Use `printf` method to format your output to match the sample runs below.

Sample run 1

N	Bubble	Selection	Insertion
5000	0.124	0.078	0.047
10000	0.453	0.280	0.173
15000	0.874	0.609	0.172
20000	1.716	1.108	0.298
25000	2.777	1.718	0.469
30000	4.181	2.559	0.687
35000	5.772	3.683	0.984
40000	8.299	5.336	1.390
45000	11.217	7.380	2.216
50000	14.634	9.860	2.622

Sample run 2

N	Bubble	Selection	Insertion
5000	0.109	0.062	0.032
10000	0.390	0.297	0.124
15000	0.875	0.611	0.181
20000	1.790	1.185	0.313
25000	2.777	1.810	0.516
30000	4.586	3.034	0.761
35000	6.568	4.103	1.203
40000	9.097	5.648	1.801
45000	12.977	8.387	2.201
50000	16.459	11.767	2.885

Grading

- 85 % for proper code structure and logic
- 15 % for correct syntax and formatting

Submission Instructions

For this assignment, you need to do the following:

1- Create a Java project of which name consists of **your student number followed by the assignment number,** e.g., "1234567_A9".
2- Create one class for each question and write your answer inside that class. Your classes should have the same name as the question number (e.g., Q1)
3- After solving all questions, open Windows Explorer (or any other file explorer).
4- Navigate to your Java project folder (can be found inside your Eclipse workspace folder).
5- Locate the "src" folder for this project (the folder that includes the source code for all questions).
6- Zip the "src" folder and rename the zipped file to match your project name (e.g., 1234567_A9.zip).
7- Submit the zipped file **to Canvas.**

Note that you can resubmit an assignment, but the new submission overwrites the old submission and receives a new timestamp.