Algorithms and Data Structures

Queues and Maps in a Corona Test Lane Simulator

Assignment-3

Version: 20.2, 8 October 2020

Introduction

In this assignment you will work on an application that predicts the throughput of Corona Test Lanes by using simulation.

As of June 1st, 2020, every Dutch citizen with some of the symptoms of CoVID-19 is advised to take a Corona test. To date, you needed to make an appointment for that by calling the GGD. But now the government wants you to provide some software by which they can predict Test Lane throughput and patient waiting times, also when citizens are advised to just go to the Test Lane without an appointment. One reason is that they also debating to give certain citizens in key professions priority of being tested and that interferes with the approach of maintaining a schedule of appointments.

To complete this assignment you will be required to select and use appropriate data structures from the categories of lists, queues, sets and maps.

Requirements

You are required to predict average and maximum waiting times of patients and workloads and finish times of nurses from one day at the Test Lane under different volumes of demand, or queuing/handling strategies which are explained below:

R1. In July 2020 there was very little exposure of Corona in the Netherlands. As a base case, you predict the performance of a Test Lane with **one nurse** and a demand profile of 100 patients across the day.

R2. In August 2020, exposure was growing, and a single nurse could not handle the demand profile of 200 patients per day anymore. You shall predict performance of **two Nurses**, working in parallel from a single (shared) queue of waiting patients.

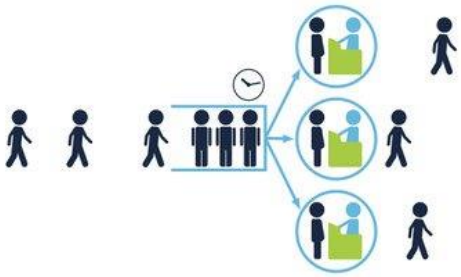

Figure 1: Shared queue service model

R3. In September 2020, exposure was growing exponentially, so the government needs a solution that can simulate **any number of nurses** at the Test Lane, serving any demand profile from a single shared queue. We wish to test your solution with 4 nurses and a demand profile of 500 patients.

R4. As of October 2020, the government want to grant citizens in key professions priority at the test-lane. There is ongoing debate about which professions are key, and they want your software to be flexible in accepting **any fraction (percentage) of priority patients** who may skip other (non-priority) patients in the waiting queue for having taken their sample. (If two patients have a same priority, then again the order of arrival at the Test Lane shall decide who goes first). We

wish to test your solution with 10% and 20% priority patients in a total demand profile of 500 patients in a Test Lane with 4 nurses.

Besides providing a sample of fluids from the nose, patients also complete a form with personal information (zipcode, date of birth, and a summary of covid related symptoms that they suffer from). This information is also registered in your application, and you are required to calculate some patient statistics from the daily demand profile (of 500 patients):

R5. The total count of patients at the Test Lane per area code (the 4 digits) of the zipcode.

R6. One zip-are code per covid symptom, which indicates the area code with the highest fraction (percentage) of patients that suffer from that symptom.
E.g.:
Assume there are 40 patients with area code '1013' of which 30 suffer from 'DRY_COUGH'
and there are 20 patients with area code '1017'of which 16 suffer from 'DRY_COUGH'
Then 80% of the patients in '1017' suffer from 'DRY_COUGH', which is a higher fraction than 75% of the patients in '1013'.
Report both the area code and the percentage of patients in your result.

(You do not need to run a simulation for these statistics, it only requires processing of patient data)

How to get going with this assignment.

Besides the requirements below also a Java/Maven starter project is provided, which includes part of the application code, realistic test data, a simulator and unit tests. You are required to complete and extend the application such that all unit tests pass, and results from any selection of test data are correct.

The starter project provides // TODO comments at every location where code is missing. In addition, you may provide additional (private) methods and/or variables/attributes as you deem fit. You should not change the signatures of public methods. You should add additional unit tests as required, but you may not change the given unit tests.

In Figure 2 you find a class diagram, which provides the design of the solution as provided in the starter project. (This design may not be complete with respect to your final solution).

The following functionality is already provided:

1. An enumeration of covid=19 symptoms that are relevant for this study
2. Definition of attributes in Patient, Nurse and TestLane classes that support tracking of relevant information for the requirements above.
3. The constructor of a Patient that support the generation of random Patient data conform some observed profile of daily demand and priority patients. The randomizer of this generator is seeded with a specific number, such that your results will be reproducible across multiple runs.
4. A formula that estimates the time it will take a nurse to complete sampling of a single patient.
5. A main class that runs the simulations for answering all of 6 requirements above.
   Here you can use different seeds for the random number generator to also run your application with different test data. With the given random seed of 19670427, the results of Figure 4 should be reproduced.

By now you may have concluded that if you solve R4 of the assignment, you also have addressed R1, R2 and R3, because R4 is a generalisation of the earlier three. But if that generalized solution does not appear immediately to you, you may choose to implement separate specific solutions for R1 and others first.

In order to determine maximum (and average) queue lengths and waiting times and the impact of prioritization, you must track all individual events of patients joining the waiting queue and being serviced by a nurse as time progresses throughout the day. The provided 'simulation algorithm' for the

test lane is provided and misses some code for gathering the statistics. The approach of that algorithm is:

A. Process all patients in the order that they arrive at the Testing Lane
B. Before a next Patient is added to the waiting queue, make sure all nurses complete all waiting work from the queue of patients that they can handle before or up till the arrival time of the next patient. That allows you to track accurately the evolution of the size of the waiting queue each time when a new patient is added.
C. Different patients will require different processing times, so the order of availability of a next nurse may vary throughout the day. Think about what the best data structure is to be able to easily track and identify the next available nurse for a next patient at the head of the waiting queue. (This is not relevant yet for the base case with only one nurse…)
D. Notice that available nurses may stand idle for a while if there are no more patients on the waiting queue. (Check that patients cannot be sampled yet at a time that they have not arrived at the Test Lane yet…)

In Figure 4 you find some of the designated outcome of this algorithm for the test data as selected by the random seed 19670427 in the main program. These results have been printed by the `TestLane.printSimulationResults()` method. That shall include the following information:

a. Some specific patient information in the format
   zipCode(*dateOfBirth*)@*arrivalTime*[*symptoms*,…]
b. For each nurse:
   - the number of patients that the nurse has sampled that day
   - the average time taken to sample a patient, with two decimals
   - the fraction of the workday that this nurse has been busy
     (total sample time / test lane open time x 100%)
c. The time that all nurses have finished all work (including overtime after closing time)
d. The maximum length of the patient's waiting queue across the day
   (Every patient first joins the queue before being invited for sampling)
e. Maximum and average wait times (with two decimals) for priority patients and regular patients.

Your report

We expect the following contents in your report:

1. A copy of the console output as it was produced by running the main application on your computer.
2. Six code snippets of the most important parts of your implementation, with clear explanations and justifications for your implementation choices.
3. Your analysis of the differences in results when applying the different priority schemes in scenario's R3 and R4 (i.e. without priority vs. 10% priority vs. 20% priority patients).

Basis of grading

Some parts of this assignment are more difficult than others…

Your solution can be sufficient if you succeed in correctly implementing any four of the six requirements.

Your solution can earn a good qualification if you succeed in correctly implementing any five of the six.

For 'excellent' grades all requirements shall be met.

See also the grading rubrics for qualifications of the grading of your report and code quality.

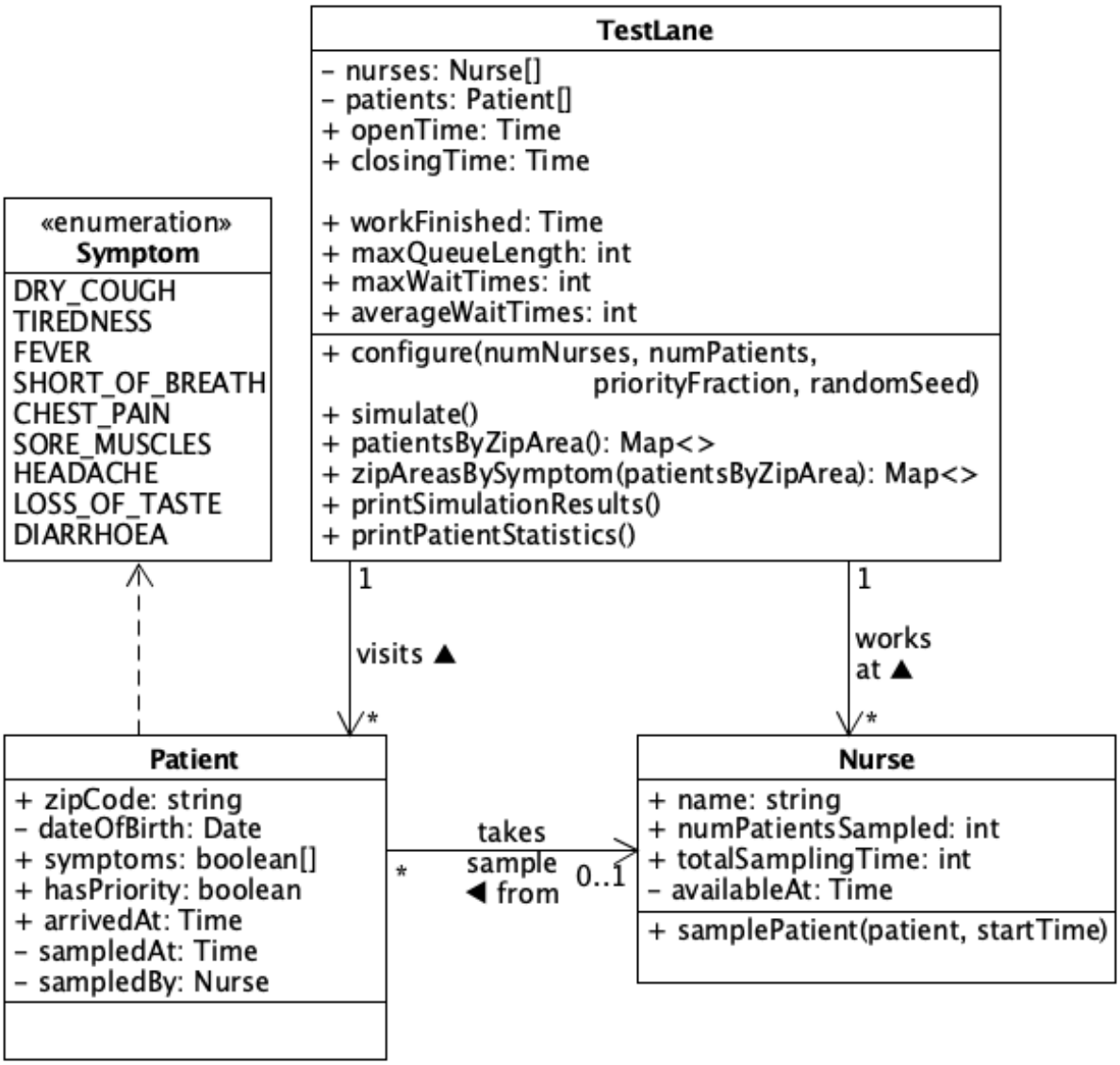

Figure 2: UML class diagram of starter project

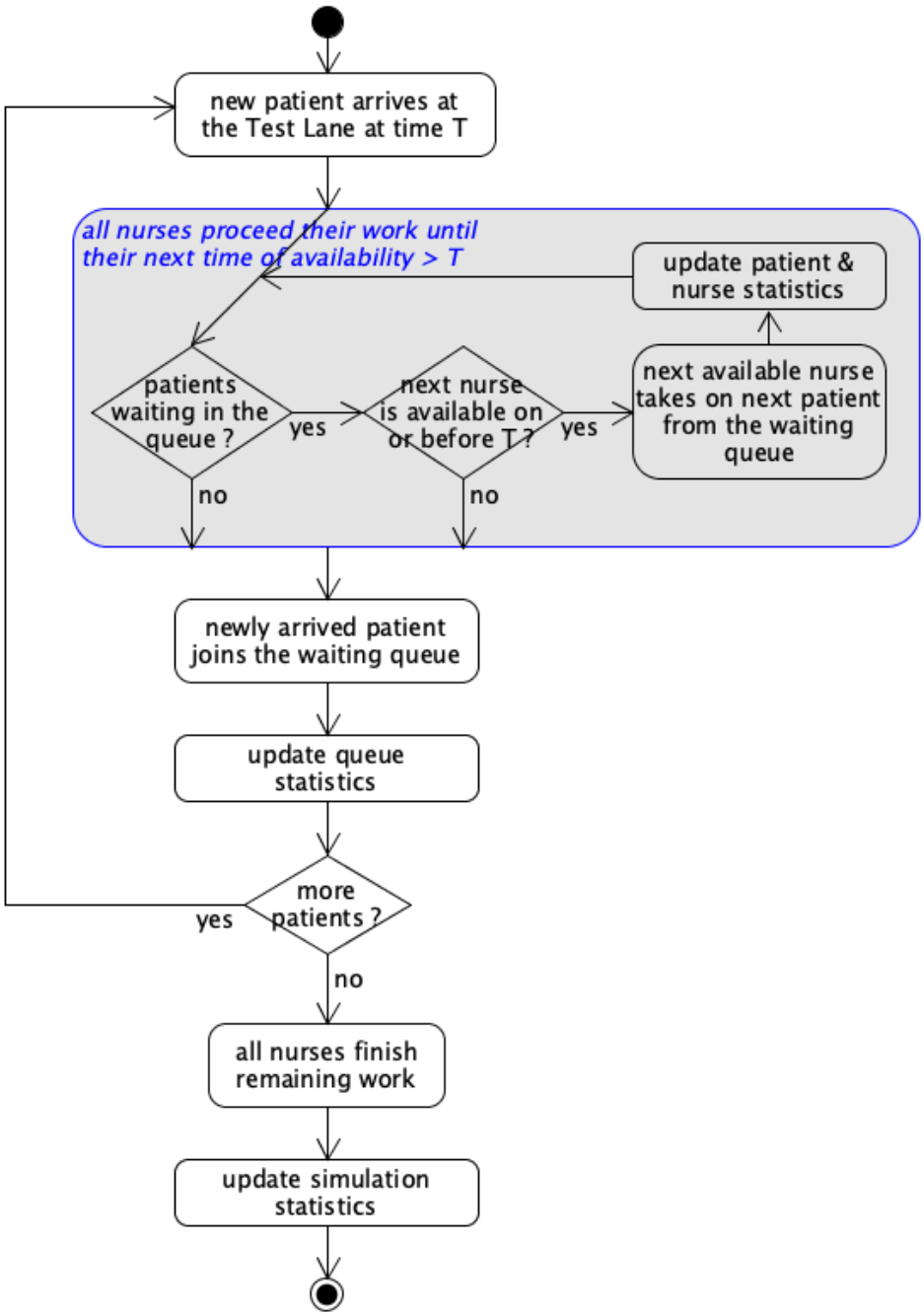

Figure 3: Proposed simulation algorithm to track patient sampling at the Test Lane

```
Corona test lane simulation between 08:00 and 16:00

Configuring test lane with 1 nurse(s) and 100 patients (0% priority); seed=19670427.
   a few patients: 1012VB(1965-02-26)@10:37:37[DRY_COUGH,FEVER,SHORT_OF_BREATH] - 1011VX(1974-04-07)
 @12:57:41[DRY_COUGH,TIREDNESS,FEVER,HEADACHE] - 1003AX(1954-07-07)@09:22:05[DRY_COUGH,FEVER] - ...
Simulating the sampling of 100 patients by 1 nurse(s).
Simulation results per nurse:
    Name: #Patients:    Avg. sample time: Workload:
 Nurse-1         100              110,53        38%
Work finished at 15:39:38
Maximum patient queue length = 6
Wait times:        Average:  Maximum:
Regular patients:    100,45       673


Configuring test lane with 2 nurse(s) and 200 patients (0% priority); seed=19670427.
   a few patients: 1012VB(1965-02-26)@10:37:37[DRY_COUGH,FEVER,SHORT_OF_BREATH] - 1011VX(1974-04-07)
 @12:57:41[DRY_COUGH,TIREDNESS,FEVER,HEADACHE] - 1003AX(1954-07-07)@09:22:05[DRY_COUGH,FEVER] - ...
Simulating the sampling of 200 patients by 2 nurse(s).
Simulation results per nurse:
    Name: #Patients:    Avg. sample time: Workload:
 Nurse-1         102              109,61        38%
 Nurse-2          98              113,96        38%
Work finished at 15:40:55
Maximum patient queue length = 4
Wait times:        Average:  Maximum:
Regular patients:     24,72       238


Configuring test lane with 4 nurse(s) and 500 patients (10% priority); seed=19670427.
   a few patients: 1012VB(1965-02-26)@10:37:37[DRY_COUGH,FEVER,SHORT_OF_BREATH] - 1011VX(1974-04-07)
 @12:57:41[DRY_COUGH,TIREDNESS,FEVER,HEADACHE] - 1003AX(1954-07-07)@09:22:05[DRY_COUGH,FEVER] - ...
Simulating the sampling of 500 patients by 4 nurse(s).
Simulation results per nurse:
    Name: #Patients:    Avg. sample time: Workload:
 Nurse-1         123              112,14        47%
 Nurse-2         125              107,86        46%
 Nurse-3         127              107,15        47%
 Nurse-4         125              110,54        47%
Work finished at 15:55:24
Maximum patient queue length = 7
Wait times:        Average:  Maximum:
Regular patients:     33,39       177
Priority patients:     9,04        58
```

Figure 4: Sample output by simulation algorithm for three scenarios