Assignment 1: Quarantine System

Note

- This is an individual assignment; all the work must be your own.
- Download the project skeleton from Canvas and finish the implementations using the skeleton.
- Upon completion, compress the directory of the project and upload a zip file to Canvas.
- The due time is 11:59 pm on March 22. Late submission will be penalized by 20% per day.
- This assignment accounts for 10% of the total course grade.
- Email to cwangch@cse.ust.hk if you have any questions.
- Cheating/plagiarism will be caught and punished HEAVILY!

University's Honor code: `http://ugadmin.ust.hk/integrity/honor.html`
Penalties for Cheating: `http://ugadmin.ust.hk/integrity/student-5.html`

1 Prerequisites, Goals, and Outcomes

1.1 Prerequisites

For the programming assignment, you need to know:

- Knowledge of class design: Class attributes, Constructors, Accessor methods, Mutator methods.
- Knowledge of inheritance: How to implement a specialization/generalization relationship using inheritance.
- Collections: Use of ArrayList class and iterators

1.2 Goals

Reinforce your ability to implement Java classes using inheritance and implement classes that use collections.

1.3 Outcomes

You will have demonstrated the following "know how to" abilities:

Class	Attribute	Meaning
Person	IDCardNo	The card ID of the person
	Loc	The location of the person, which is a pair of integers
	Gender	Male or female
	Age	The age of the person
	IsVaccinated	Whether the person has been vaccinated
	InfectCnt	The count of being infected
Patient	SymptomLevel	Critical, Moderate, or Mild
	HospitalID	The ID of the hospital which provides the treatment
Hospital	HospitalID	The identity of the hospital
	Loc	The location of the hospital in the x-axis and y-axis
	Cap	The capacities of three kinds of medical treatment
	CriticalPatients	The lists of critically-ill patients in the hospital
	ModeratePatients	The lists of moderately-ill patients in the hospital
	MildPatients	The lists of mild patients in the hospital
Record	IDCardNo	The card ID of the patient
	symptomLevel	The symptom level of the patient
	status	Whether the patient is confirmed or recovered
	HospitalID	The ID of the hospital providing the treatment
Dashboard	patientNums	The numbers of patients in different age ranges
	infectAvgNums	The average numbers of infected counts
	vacNums	The numbers of vaccinated people
	vacInfectNums	The numbers of vaccinated patient
Quarantine System	people	The list of all the people
	patients	The list of all the patients
	hospitals	The list of all the hospitals
	records	The list of all the records
	dashboard	The statistics of the quarantine

Table 1: Classes in the quarantine system.

- Implement a Java class and its constructors, accessors, and the mutators.
- Use inheritance to implement specialization/generalization relationships.
- Implement a Java class that uses collections.

2 Problem Description

You are going to implement a simulation system in the pandemic situation. The objective of the system is to fight the spread of the Covid19 virus and design the quarantine strategy for each patient. There are five major entities in the system, which are listed in Table 1. Particularly, the dashboard contains the statistics you need to collect in the quarantine, which will be described in Task 2 as below. To help you solve the problem, we provide the

implementation of most of the methods in the above classes. Also, we have provided the full implementation of the classes of Location and Capacity to wrap the integers as more abstract program constructs. We hope they can help you define the rest of the implementation.

Initially, all the people are not infected, and there are several existing hospitals. Then the patients are confirmed and recovered continuously, of which the information is maintained in the attribute named Records in the quarantine system. As a designer of the system, you are required to process each record to determine how to save each patient by sending him/her to a hospital, or releasing a patient who has been recovered. Please try your best to help the patients and finish the following two missions.

2.1 Task 1: Saving Patients

For each record, try to process it based on its status.

Confirmed Case: Find the nearest hospital which still has the capacity of providing the corresponding kind of treatment. You can just adopt the Euclidean distance to measure the distance of two locations. If there does not exist the available hospital, establish a new hospital at the location of the patient, of which CriticalCapacity, ModerateCapacity, and MildCapacity are 5, 10, and 20, respectively. The identity of the n-th new hospital should be "H-New-n". For example, the first new hospital should have "H-New-1" as its identity.

Recovered Case: Remove the patient from the patient list of the hospital, and increase the corresponding capacity.

Please remember updating the attribute InfectCnt to maintain the count of being infected for each patient, which is necessary for collecting the statistics in the next task. Particularly, you should update the InfectCnt once the patient has been confirmed to be positive. If the patient is not recovered according to the records, you still need to count the last infection.

2.2 Task 2: Collecting Statistics

To support further medical research and decision, you are expected to summarize the statistics of the people in different age ranges, including $(0, 10)$, $[10, 20)$, $[20, 30)$, $[30, 40)$, $[40, 50)$, $[50, 60)$, $[60, 70)$, $[70, \infty)$. The information you need to collect is listed as follows.

- The numbers of the people who are infected at least one time in different age ranges.
- The average counts of being infected in different age ranges. Please note that you should take the number of the patients in a specific age range as the divisor rather than the total number of the people.
- The numbers of vaccinated people in different age ranges.
- The numbers of vaccinated but still infected people in different age ranges.

3 Input and Output

3.1 Input

We provide three tables, named Person, Hospital, and Record.

- Person contains the columns named IDCardNo, XLoc, YLoc, gender, age, and isVac.
- Hospital contains the columns named HospitalID, XLoc, YLoc, Critical, Moderate, and Mild. The last three columns maintains the capacities of treating three kinds of the patients.
- Record contains the columns named IDCardNo, SymptomLevel, and Status.

You should handle the records according to their orders on the table Record. Assume that all the cells in the tables are in valid forms. The existing hospitals in hospital.xlsx have the identities in different forms of "HOldn", assuring that their identities are different from the ones of the new hospitals. Also, the records assure that each patient can only be infected again after the recovery, and similarly, a patient can only recover after being infected. That is, there do not exist two records of the same person with the status *Confirmed*/*Recovered* without a record of the person with the status *Recovered*/*Confirmed* between them.

To make things easier, we provide the three tables in the txt files, so you do not need to import any three-party libraries to process xlsx, csv, or json files. We have provided the implementations of importing and exporting the data, and you can invoke them directly.

Lastly, the numbers of the rows in Person, Hospital, and Record are not larger than 100,000, 10, and 10,000, respectively.

3.2 Output

You are expected to generate the txt file named RecordTreatment.txt in the first task, which appends another column named HospitalID to the table Record in Record.txt, showing the hospital providing the medical treatment for the patient.

After finishing the first task, you are expected to generate the file named Statistics.txt for the second task, which contains a collection of 5-tuples. The first entry is the age range, and the last four entries are the four quantities defined in the task description. All the rows should be sorted according to the age range, i.e., the first row corresponds to the statistics of $(0, 10)$, and the last row corresponds to the statistics of $[70, \infty)$.

4 Tips

In total, there are eight `TODO` annotations in the skeleton. Basically, you are only required to finish the parts labeled with the `TODO` annotations, and simply utilize the implementations we have provided in the skeleton for other parts. However, it is also encouraged to implement each part by yourself and try to optimize the efficiency of each method. You can even change the definitions of the classes and redesign the system as you like, but you should make sure

your code can always generate the expected results. Particularly, you should not change the functionality of the methods for data exporting even if you do the refactoring.

To help you debug and test your implementation, we also provide a test suite containing three test cases in the skeleton, which are stored in the directory named sampleData. You can put the input files to the directory named data in the project root directory and generate the output files in the directory named output after executing your program. Before the submission, please make sure that your implementation can generate the same results for each input sample.

5 Bonus

Apart from the scores for the correct implementation, we also assign the bonus to the implementation with high efficiency. Totally, we have ten test cases for the assignment grading, and you can get one score if your implementation finishes the two task for one test case in one second. Therefore, the upper bound of the bonus is ten scores, and the full mark is 110. You can try your best to use better data structures, such as map and set in Java Collections Framework, to obtain good performance.