### Disease Outbreak Response Management System (DORMS)

#### CS2030/S Mock Practical Assessment I

**Topic Coverage**

* Classes, Methods, Attributes and Methods
* Abstraction, Encapsulation, Inheritance and Polymorphism
* Collections
* Abstract Classes
* SOLID principles
* Generics

### Problem Description

We are in the midst of a global crisis. SARS-CoV-2 (SARS CoronaVirus 2), the virus that causes COVID-19 has infected millions of people worldwide and killed thousands.

In Singapore, we have implemented two systems, SafeEntry and TraceTogether. Right now, you are a junior developer in the team that is developing the Disease Outbreak Response Management System (DORMS), that brings both of these systems together. In addition, DORMS will automatically issue Stay-Home Notices (similar to a quarantine order).

The actual system needs a lot of time to develop. You're in charge of completing a simulation of DORMS, which simulates how DORMS will operate when it has been implemented, and keeps track of statistics and the efficacy of the policies enacted.

The simulation environment of DORMS is extremely complex. As such, read the following explanations carefully.

### The simulation

#### Viruses

In the simulation, there are three types of viruses. All viruses spread by creating another virus of the same type, or another virus of a different type (mutation). Here is how each virus spreads:

* AlphaCoronavirus: Every time it spreads, it has some probability of mutating into SARS-CoV-2. In the event that it doesn't mutate, the virus will simply create another AlphaCoronavirus, but the probability of that new virus mutating is reduced by 10% (given by SimulationParameters#VIRUS\_MUTATION\_PROBABILITY\_REDUCTION).
* SARS-CoV-2: This is the target virus of the simulation, and this causes COVID-19. Every time it spreads, it has some probability of mutating into a BetaCoronavirus. In the event that it doesn't mutate, the probability of that new virus mutating is reduced by 10%.
* BetaCoronavirus: This virus has no probability of mutating. As such, every time it spreads, it simply creates another BetaCoronavirus.

#### People

Of course, people are the key of this simulation, who are the primary vectors of the viruses above. People can transmit (give out) viruses they have, and can be infected with (take in) viruses from a contact. This simulation concerns itself with two types of people:

* Person: Represents the average person like you and me. When transmitting viruses, the person will transmit all the viruses he/she is infected with.
* MaskedPerson: This person is wearing a mask. In the simulation, masks are 60% effective (given by SimulationParameters#MASK\_EFFECTIVENESS). This means that there is a 60% chance that at any given contact, no viruses will be transmitted. Likewise, there is a 60% chance that at any given contact, this person will not be infected with any virus.

#### DORMS

As a junior developer, you are not required to implement the full solution yourself. The higher level classes have already been implemented for you. All you need to do is to implement the concrete classes that represent different entities of the simulation. However, for clarity, the following explains how DORMS works and provides some specifications for you as you complete the implementation.

DORMS is essentially a combination of two existing solutions:

* SafeEntry: This is a system that allows users to check in and out of different locations. This allows us to keep track of all the contacts made in a location, to prevent and control the transmission of diseases and identify disease clusters. With SafeEntry, it is assumed that a person entering a location makes contact with everyone in that location, and diseases are spread via each contact.
* TraceTogether: This is a programme to enhance Singapore's contact tracing efforts. The TraceTogether app is a mobile application that uses bluetooth to detect other nearby TraceTogether-enabled devices. On device detection, it is assumed that contact is made and diseases are spread via that contact.

Given these two systems, DORMS is a single platform that interacts with both of these systems for contact tracing. It allows users to check in to a location (via checkIn), check out of a location (via checkOut), keeps track of other contacts for TraceTogether (via contact). It also notes any person who presents with symptoms of respiratory illnesses (via presentSymptoms). DORMS will conduct a serological test on anyone who presents themselves with these symptoms, and take the necessary action if the person tests positive for the target virus (SARS-CoV-2). The following describes the action that needs to be taken by DORMS on a positive SARS-CoV-2 test:

1. The person who tested positive will be given a 28-day Stay-Home-Notice and will not be allowed to leave their home during that period.
2. All recent (14 days, or SimulationParameters#TRACING\_PERIOD) contacts made by this person will be served a 14-day (or SimulationParameters#SHN\_DURATION) Stay-Home-Notice as well.

However, the current implementation of DORMS does not support it, but thanks to SOLID principles, the developers of DORMS have made it open for extension, and it will be simple to implement this behaviour.

### Your Task

You are given the incomplete implementation of the DORMS simulation. Your task is to complete the implementation of the missing classes.

After completing the program, you may run the DORMS simulation:

$ java Main  
===== RUNNING SIMULATION =====  
Mask policy not implemented and SHN not issued  
  
===== STATISTICS =====  
Infected population: 11  
Total Population: 19  
===== SIMULATION COMPLETED =====  
  
===== RUNNING SIMULATION =====  
Mask policy not implemented and SHN issued  
  
===== STATISTICS =====  
Infected population: 6  
Total Population: 19  
===== SIMULATION COMPLETED =====  
  
===== RUNNING SIMULATION =====  
Mask policy implemented and SHN not issued  
  
===== STATISTICS =====  
Infected population: 7  
Total Population: 19  
===== SIMULATION COMPLETED =====  
  
===== RUNNING SIMULATION =====  
Mask policy implemented and SHN issued  
  
===== STATISTICS =====  
Infected population: 4  
Total Population: 19  
===== SIMULATION COMPLETED =====

Or you can run the simulation in verbose mode by issuing the -v flag, such as by running java Main -v

This task is divided into several levels. Read through all the levels to see how the different levels are related. You are to complete **ALL** levels.

All the files given to you should not be modified. You may modify them for your own testing, but we will replace them to conduct our own tests on your solutions.

| Level 1 **Creating the immutable Virus class**  The first thing we'd define is the Virus class, because most of the entities of this simulation relies on its implementation.  We are going to define a Virus class where each virus will inherit from. This follows the Open-Closed and Liskov Substitution Principles nicely, and subsequently, you'll observe that these will come in handy when viruses are being spread for person to person.  The Virus class will have attributes depending on your implementation. However, your supervisor has instructed you ensure that your Virus class has the following methods that adhere to the following specifications:   * Virus spread(double random). This method essentially causes the virus to spread, returning a new virus. It takes in a random value as a double. Essentially, if random <= probabilityOfMutating, then the virus mutates. * boolean test(String name). This method essentially checks if this virus has name as its name. The method will be used for the serological tests conducted. * The string representation of virus objects is shown in the jshell output   Note that by SOLID principles, you should have aimed to keep Virus abstract and implement concrete AlphaCoronavirus, BetaCoronavirus and SARS\_CoV\_2 classes. Note the naming, especially with SARS\_CoV\_2 where we use a small 'o'. Note that the name of the virus is spelt with - instead of \_; we use \_ as the name of the class instead, for obvious reasons.   | $ jshell -q your\_files\_in\_ascending\_dependency\_order < test1.jsh jshell> new AlphaCoronavirus(1).spread(0) $.. ==> SARS-CoV-2 with 1.000 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.51) $.. ==> Alpha Coronavirus with 0.450 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.5) $.. ==> SARS-CoV-2 with 0.500 probability of mutating jshell> new AlphaCoronavirus(1).spread(0).spread(1) $.. ==> Beta Coronavirus with 0.000 probability of mutating jshell> new AlphaCoronavirus(1).spread(0).spread(0.5) $.. ==> Beta Coronavirus with 0.000 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.51) $.. ==> Alpha Coronavirus with 0.450 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.51).spread(0.4) $.. ==> SARS-CoV-2 with 0.450 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.5) $.. ==> SARS-CoV-2 with 0.500 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.5).spread(0.6) $.. ==> SARS-CoV-2 with 0.450 probability of mutating jshell> new AlphaCoronavirus(0.1).spread(0).spread(1).spread(1).spread(1)  $.. ==> SARS-CoV-2 with 0.073 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.5).spread(0.6) $.. ==> SARS-CoV-2 with 0.450 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.5).spread(0.6).spread(0.46) $.. ==> SARS-CoV-2 with 0.405 probability of mutating jshell> new AlphaCoronavirus(0.5).spread(0.5).spread(0.6).spread(0.45) $.. ==> Beta Coronavirus with 0.000 probability of mutating jshell> new SARS\_CoV\_2(0.3) $.. ==> SARS-CoV-2 with 0.300 probability of mutating jshell> new SARS\_CoV\_2(0.3).spread(0.3) $.. ==> Beta Coronavirus with 0.000 probability of mutating jshell> new BetaCoronavirus() $.. ==> Beta Coronavirus with 0.000 probability of mutating jshell> /exit | | --- | |
| --- | --- |
| Level 2 **Representing People**  Now, we are going to implement a **mutable** Person class.  These are the specifications for the Person class:   * transmit(double random) is a method that returns the list of the viruses to transmit after spreading, where random is used to determine if each virus mutates or not. You should be able to infer the return type. * void infectWith(SomeType listOfViruses, double random). This is a method that causes the person to be infected with the listOfViruses. You should be able to determine what SomeType is. Note that for this method, the random value is insignificant because the viruses do not mutate before entering the human body, but only when exiting. * The String representation and constructor arguments can be inferred from the following jshell tests.  | $ jshell -q your\_files\_in\_ascending\_dependency\_order < test2.jsh jshell> Person illio = new Person("Illio"); jshell> Person phillmont = new Person("Phillmont") jshell> Person nicole = new Person("Nicole") jshell> Person shuming = new Person("Shuming") jshell> illio.infectWith(List.of(new AlphaCoronavirus(1)), 0); jshell> phillmont.infectWith(List.of(new AlphaCoronavirus(0.5)), 0); jshell> nicole.infectWith(List.of(new SARS\_CoV\_2(0.5), new AlphaCoronavirus(0.4)), 0); jshell> shuming.infectWith(List.of(new AlphaCoronavirus(0.1)), 0); jshell> Arrays.toString(illio.transmit(1).toArray()) $.. ==> "[SARS-CoV-2 with 1.000 probability of mutating]" jshell> Arrays.toString(illio.transmit(1).toArray()) $.. ==> "[SARS-CoV-2 with 1.000 probability of mutating]" jshell> Arrays.toString(phillmont.transmit(0.51).toArray()) $.. ==> "[Alpha Coronavirus with 0.450 probability of mutating]" jshell> Arrays.toString(phillmont.transmit(0.5).toArray()) $.. ==> "[SARS-CoV-2 with 0.500 probability of mutating]" jshell> Arrays.toString(nicole.transmit(0.51).toArray()) $.. ==> "[SARS-CoV-2 with 0.450 probability of mutating, Alpha Coronavirus with 0.360 probability of mutating]" jshell> Arrays.toString(nicole.transmit(0.5).toArray()) $.. ==> "[Beta Coronavirus with 0.000 probability of mutating, Alpha Coronavirus with 0.360 probability of mutating]" jshell> Arrays.toString(nicole.transmit(0.4).toArray()) $.. ==> "[Beta Coronavirus with 0.000 probability of mutating, SARS-CoV-2 with 0.400 probability of mutating]" jshell> List l = illio.transmit(1) jshell> shuming.infectWith(l, 1); jshell> Arrays.toString(shuming.transmit(1).toArray()) $.. ==> "[Alpha Coronavirus with 0.090 probability of mutating, Beta Coronavirus with 0.000 probability of mutating]" jshell> Arrays.toString(shuming.transmit(0).toArray()) $.. ==> "[SARS-CoV-2 with 0.100 probability of mutating, Beta Coronavirus with 0.000 probability of mutating]" jshell> illio.test("Alpha Coronavirus") $.. ==> true jshell> illio.test("SARS-CoV-2") $.. ==> false jshell> illio.test("Beta Coronavirus") $.. ==> false jshell> phillmont.test("Alpha Coronavirus") $.. ==> true jshell> nicole.test("SARS-CoV-2") $.. ==> true jshell> shuming.test("Beta Coronavirus") $.. ==> false jshell> phillmont.getName() $.. ==> "Phillmont" jshell> /exit | | --- | |
| Level 3 **Making Contact**  Now we are going to get two people to contact each other and spread viruses.  Create an **immutable** Contact class that keeps track of a contact between two people, and transmits viruses between them. Note that the viruses should be transmitted immediately upon the instantiation of the Contact object.  The following are the specifications of the Contact class:   * Contact(Person first, Person second, double time). This is the constructor which keeps the references of the two people in contact, while simultaneously transmitting viruses between each other. Note that the transmission and infection of viruses happen in parallel, meaning that a infects b at the same time that b infects a. Also, the random value should be determined by RandomNumberGenerator (see RandomNumberGenerator#nextDouble). * boolean involves(Person p, double time). This method checks if the Contact involves p, and happened after time time. * Person other(Person p). This method simply returns the other Person involved in the contact. Assume that p will always be a valid member of the contact. * double timeOfContact(). This method simply returns the time of contact   Remember that cyclic dependencies are not allowed.   | $ jshell -q your\_files\_in\_ascending\_dependency\_order < test3.jsh jshell> RandomNumberGenerator.reset() jshell> RandomNumberGenerator.nextDouble() $.. ==> 0.6182744543846053 jshell> RandomNumberGenerator.reset() jshell> Person daekoon = new Person("Daekoon") jshell> Person junda = new Person("Jun Da") jshell> Person ziyang = new Person("Zi Yang") jshell> daekoon.infectWith(List.of(new AlphaCoronavirus(0.99)), 1); jshell> Contact c = new Contact(daekoon, junda, 1); jshell> daekoon.transmit(1) $.. ==> [Alpha Coronavirus with 0.891 probability of mutating] jshell> daekoon.transmit(0) $.. ==> [SARS-CoV-2 with 0.990 probability of mutating] jshell> junda.transmit(1) $.. ==> [SARS-CoV-2 with 0.891 probability of mutating] jshell> junda.transmit(0) $.. ==> [Beta Coronavirus with 0.000 probability of mutating] jshell> c.involves(daekoon, 1) $.. ==> true jshell> c.involves(daekoon, 1.1) $.. ==> false jshell> c.involves(ziyang, 0) $.. ==> false jshell> c.other(daekoon) $.. ==> Jun Da jshell> c.other(junda) $.. ==> Daekoon jshell> c.timeOfContact() $.. ==> 1.0 jshell> RandomNumberGenerator.reset() jshell> Person junda = new Person("Jun Da") jshell> Person ziyang = new Person("Zi Yang") jshell> junda.infectWith(List.of(new SARS\_CoV\_2(0.5)), 1); jshell> Contact d = new Contact(ziyang, junda, 1); jshell> ziyang.transmit(1) $.. ==> [SARS-CoV-2 with 0.405 probability of mutating] jshell> ziyang.transmit(0) $.. ==> [Beta Coronavirus with 0.000 probability of mutating] jshell> junda.transmit(1) $.. ==> [SARS-CoV-2 with 0.450 probability of mutating] jshell> junda.transmit(0) $.. ==> [Beta Coronavirus with 0.000 probability of mutating] jshell> d.timeOfContact() $.. ==> 1.0 jshell> /exit | | --- |   Note that Dogs cannot eat Chocolate, and hence A does not eat the Chocolate. Cats also cannot eat cheese. In both cases, both A and Meowie stop eating as they are full. |
| Level 4 **Locations**  So far, we've only dealt with contact tracing, and have yet to deal with SafeEntry's cluster tracking. DORMS has specified a **mutable** Location class that keeps track of its occupants at any given time.  The following defines the specifications for the Location class:   * getOccupants. This essentially returns a list of all the occupants in the Location. You can infer the return type. * void accept(Person person). This accepts a Person into the Location. * void remove(Person person). This removes a Person from this Location. * The String representation and constructor arguments can be inferred from the jshell output.  | $ jshell -q your\_files\_in\_ascending\_dependency\_order < test4.jsh jshell> Person mingsoon = new Person("Ming Soon") jshell> Person longThePerson = new Person("Long") jshell> Person guoyu = new Person("Guoyu") jshell> Person rishabh = new Person("Rishabh") jshell> Person nigel = new Person("Nigel") jshell> Location l = new Location("LT19") jshell> Arrays.toString(l.getOccupants().toArray()) $.. ==> "[]" jshell> l.accept(mingsoon) jshell> l.accept(longThePerson) jshell> Arrays.toString(l.getOccupants().toArray()) $.. ==> "[Ming Soon, Long]" jshell> l.accept(guoyu) jshell> l.remove(mingsoon) jshell> Arrays.toString(l.getOccupants().toArray()) $.. ==> "[Long, Guoyu]" jshell> l.accept(rishabh) jshell> l.accept(nigel) jshell> Arrays.toString(l.getOccupants().toArray()) $.. ==> "[Long, Guoyu, Rishabh, Nigel]" jshell> l.remove(longThePerson) jshell> l.remove(guoyu) jshell> l.remove(rishabh) jshell> l.remove(nigel) jshell> Arrays.toString(l.getOccupants().toArray()) $.. ==> "[]" jshell> /exit | | --- | |
| Level 5 **Mask Policy**  We are aiming to simulate the efficacy of a mask-wearing policy. As such, we need some way to represent the behaviour of people wearing masks.  Hopefully, you have kept your Person class open for extension. As such, we can quite simply extend from the Person class to create a MaskedPerson class, which follows the same specifications.  Note that for both transmissions and infections, if the random value supplied is less than or equal to the mask's effectiveness (see SimulationParameters#MASK\_EFFECTIVENESS), then nothing is transmitted / infected.  The remaining specifications can be inferred from the jshell test. Note that you should adhere to the DRY (Don't-Repeat-Yourself) principle as much as possible. Expect to make calls to super in your overriden methods.   | $ jshell -q your\_files\_in\_ascending\_dependency\_order < test5.jsh jshell> MaskedPerson gaogui = new MaskedPerson("Gao Gui") jshell> MaskedPerson yuchen = new MaskedPerson("Yu Chen") jshell> MaskedPerson gabriel = new MaskedPerson("Gabriel") jshell> MaskedPerson yuze = new MaskedPerson("Yuze") jshell> MaskedPerson cheehong = new MaskedPerson("Chee Hong") jshell> gaogui.infectWith(List.of(new AlphaCoronavirus(0.99)), 0.61) jshell> Arrays.toString(gaogui.transmit(1).toArray()) $.. ==> "[Alpha Coronavirus with 0.891 probability of mutating]" jshell> Arrays.toString(gaogui.transmit(0.99).toArray()) $.. ==> "[SARS-CoV-2 with 0.990 probability of mutating]" jshell> Arrays.toString(gaogui.transmit(0.6).toArray()) $.. ==> "[]" jshell> gaogui.infectWith(List.of(new SARS\_CoV\_2(1)), 0.6) jshell> Arrays.toString(gaogui.transmit(1).toArray()) $.. ==> "[Alpha Coronavirus with 0.891 probability of mutating]" jshell> Arrays.toString(gaogui.transmit(0.99).toArray()) $.. ==> "[SARS-CoV-2 with 0.990 probability of mutating]" jshell> Arrays.toString(gaogui.transmit(0.6).toArray()) $.. ==> "[]" jshell> yuchen.infectWith(List.of(new SARS\_CoV\_2(1), new AlphaCoronavirus(0.5)), 0.7); jshell> Arrays.toString(yuchen.transmit(0.61).toArray()) $.. ==> "[Beta Coronavirus with 0.000 probability of mutating, Alpha Coronavirus with 0.450 probability of mutating]" jshell> Arrays.toString(yuchen.transmit(0.6).toArray()) $.. ==> "[]" jshell> gabriel.infectWith(List.of(new AlphaCoronavirus(0.5)), 0.61) jshell> RandomNumberGenerator.reset() jshell> Contact c = new Contact(gabriel, yuze, 1) jshell> c = new Contact(gabriel, cheehong, 1) jshell> c = new Contact(yuze, cheehong, 1) jshell> Arrays.toString(gabriel.transmit(0.61).toArray()) $.. ==> "[Alpha Coronavirus with 0.450 probability of mutating]" jshell> Arrays.toString(gabriel.transmit(0.6).toArray()) $.. ==> "[]" jshell> Arrays.toString(yuze.transmit(0.61).toArray()) $.. ==> "[Alpha Coronavirus with 0.405 probability of mutating]" jshell> Arrays.toString(yuze.transmit(0.6).toArray()) $.. ==> "[]" jshell> Arrays.toString(cheehong.transmit(0.61).toArray()) $.. ==> "[]" jshell> Arrays.toString(cheehong.transmit(0.6).toArray()) $.. ==> "[]" jshell> /exit | | --- | |
| Level 6 **Stay-Home-Notice (SHN) Policy**  We are aiming to simulate the efficacy of an SHN policy. Because DORMS currently does not support this, we need to extend the Dorms class to implement this behaviour.  Because we need to keep track of SHNs being given out, we can use the Person class to maintain the state of each Person, i.e. whether he/she is on SHN. This results in two new methods required in the class:   * boolean serveSHN(double shnEndTime). This issues the person an SHN that ends at shnEndTime. The method returns true only if an SHN was served; if Person A is serving an SHN until time t = 10, and is now issued a second SHN that ends at time t = 9, then the second SHN is not issued as it makes no difference. * boolean onSHN(double currentTime). This takes in the current time and returns true if the Person is on SHN.   Your job now is to implement the DormsWithShn class which is basically DORMS + automatic issuance of SHNs. Remember that you are not allowed to modify the Dorms class. Many methods were purposefully declared final to prevent extension.  Read the remaining classes provided to you carefully. In particular, here are some tips to help you implement the feature:   * The main logic you need to override is handleSickPerson. You do not need to override other methods or define other behaviours. * The constructor for DormsWithShn class should be the same as Dorms. * Make use of the queryContacts method. This retrieves all the Contacts that are related to the sick Person. This also queries the Contacts up to SimulationParameters#TRACING\_PERIOD in history. * Remember to log every SHN served. The format of the output can be seen in the verbose output of the simulation below. Do not use System.out.println(), but use log instead. * The behaviour of people checking in or making contact while on SHN is well defined. You do not need to implement this behaviour.   Once you're done with the implementation, you may proceed to run the DORMS simulation by compiling Main.java and running Main on the JVM. You may also supply the -v flag for verbose mode, which shows all your logs.  We hope you enjoyed this mock PE, best of luck for next Friday :)  One final question, do you think DORMS was well designed? How can we improve the design of DORMS?  - CS2030/S Teaching Team   | $ java Main ===== RUNNING SIMULATION ===== Mask policy not implemented and SHN not issued  ===== STATISTICS ===== Infected population: 11 Total Population: 19 ===== SIMULATION COMPLETED =====  ===== RUNNING SIMULATION ===== Mask policy not implemented and SHN issued  ===== STATISTICS ===== Infected population: 6 Total Population: 19 ===== SIMULATION COMPLETED =====  ===== RUNNING SIMULATION ===== Mask policy implemented and SHN not issued  ===== STATISTICS ===== Infected population: 7 Total Population: 19 ===== SIMULATION COMPLETED =====  ===== RUNNING SIMULATION ===== Mask policy implemented and SHN issued  ===== STATISTICS ===== Infected population: 4 Total Population: 19 ===== SIMULATION COMPLETED =====    $ java Main -v ===== RUNNING SIMULATION ===== Mask policy not implemented and SHN not issued  Initial Disease Carrier visits LT19 at time 0.000 Prof Henry visits LT19 at time 0.000 Prof Terence visits LT19 at time 0.000 Yong Qi visits LT19 at time 0.100 Kevin visits LT19 at time 0.110 Prof Henry leaves LT19 at time 0.200 Sean visits LT19 at time 0.230 Yong Qi leaves LT19 at time 0.300 Yong Qi met Eric at time 0.330 Eric met De Zhang at time 0.340 Prof Terence leaves LT19 at time 0.400 Prof Henry visits i3 at time 0.450 Initial Disease Carrier leaves LT19 at time 0.500 Kevin leaves LT19 at time 0.500 Sean leaves LT19 at time 0.500 Yong Qi tests positive for SARS-CoV-2 at time 1.000 De Zhang visits i3 at time 1.100 Sean visits i3 at time 1.200 Initial Disease Carrier visits i3 at time 1.300 De Zhang leaves i3 at time 1.300 Initial Disease Carrier leaves i3 at time 1.300 Sean leaves i3 at time 1.300 Prof Henry leaves i3 at time 1.300 Prof Henry met De Zhang at time 1.300 Eric met Prof Henry at time 1.400 De Zhang tests positive for SARS-CoV-2 at time 3.000 Initial Disease Carrier visits COM1-B113 at time 3.100 Marcus visits COM1-B113 at time 3.100 Jerryl visits COM1-B113 at time 3.200 Yong Qi visits COM1-B113 at time 3.300 Prof Henry visits COM1-B113 at time 3.400 Kevin visits COM1-B114 at time 3.500 Destinee visits COM1-B114 at time 3.600 Jerryl tests positive for SARS-CoV-2 at time 3.600 Yong Qi leaves COM1-B113 at time 3.600 Initial Disease Carrier leaves COM1-B113 at time 4.000 Marcus leaves COM1-B113 at time 4.000 Jerryl leaves COM1-B113 at time 4.000 Prof Henry leaves COM1-B113 at time 4.000 Kevin leaves COM1-B114 at time 4.000 Destinee leaves COM1-B114 at time 4.000 Siddarth Raj met Yong Qi at time 5.000 Xuan Ming met Yong Qi at time 7.000 Xuan Ming visits ION Orchard at time 9.000 Jerryl visits ION Orchard at time 9.400 Xuan Ming leaves ION Orchard at time 9.500 Xuan Ming met Le Yang at time 10.000 Le Yang met Joel at time 10.500 Mario met Jeremy at time 10.500 Bryan visits ION Orchard at time 10.500 Geyu visits ION Orchard at time 10.500 Jerryl leaves ION Orchard at time 10.500 Geyu leaves ION Orchard at time 10.500 Bryan leaves ION Orchard at time 10.500 Mario test negative for SARS-CoV-2 at time 11.000 ===== STATISTICS ===== Infected population: 11 Total Population: 19 ===== SIMULATION COMPLETED =====  ===== RUNNING SIMULATION ===== Mask policy not implemented and SHN issued  Initial Disease Carrier visits LT19 at time 0.000 Prof Henry visits LT19 at time 0.000 Prof Terence visits LT19 at time 0.000 Yong Qi visits LT19 at time 0.100 Kevin visits LT19 at time 0.110 Prof Henry leaves LT19 at time 0.200 Sean visits LT19 at time 0.230 Yong Qi leaves LT19 at time 0.300 Yong Qi met Eric at time 0.330 Eric met De Zhang at time 0.340 Prof Terence leaves LT19 at time 0.400 Prof Henry visits i3 at time 0.450 Initial Disease Carrier leaves LT19 at time 0.500 Kevin leaves LT19 at time 0.500 Sean leaves LT19 at time 0.500 Yong Qi tests positive for SARS-CoV-2 at time 1.000 Initial Disease Carrier has been served a SHN that ends at 14.100 Prof Henry has been served a SHN that ends at 14.100 Prof Terence has been served a SHN that ends at 14.100 Kevin has been served a SHN that ends at 14.110 Sean has been served a SHN that ends at 14.230 Eric has been served a SHN that ends at 14.330 De Zhang visits i3 at time 1.100 De Zhang leaves i3 at time 1.300 De Zhang tests positive for SARS-CoV-2 at time 3.000 Eric has been served a SHN that ends at 14.340 Prof Henry has been served a SHN that ends at 15.100 Marcus visits COM1-B113 at time 3.100 Jerryl visits COM1-B113 at time 3.200 Destinee visits COM1-B114 at time 3.600 Jerryl test negative for SARS-CoV-2 at time 3.600 Marcus leaves COM1-B113 at time 4.000 Jerryl leaves COM1-B113 at time 4.000 Destinee leaves COM1-B114 at time 4.000 Xuan Ming visits ION Orchard at time 9.000 Jerryl visits ION Orchard at time 9.400 Xuan Ming leaves ION Orchard at time 9.500 Xuan Ming met Le Yang at time 10.000 Le Yang met Joel at time 10.500 Mario met Jeremy at time 10.500 Bryan visits ION Orchard at time 10.500 Geyu visits ION Orchard at time 10.500 Jerryl leaves ION Orchard at time 10.500 Geyu leaves ION Orchard at time 10.500 Bryan leaves ION Orchard at time 10.500 Mario test negative for SARS-CoV-2 at time 11.000 ===== STATISTICS ===== Infected population: 6 Total Population: 19 ===== SIMULATION COMPLETED =====  ===== RUNNING SIMULATION ===== Mask policy implemented and SHN not issued  Initial Disease Carrier (masked) visits LT19 at time 0.000 Prof Henry (masked) visits LT19 at time 0.000 Prof Terence (masked) visits LT19 at time 0.000 Yong Qi (masked) visits LT19 at time 0.100 Kevin (masked) visits LT19 at time 0.110 Prof Henry (masked) leaves LT19 at time 0.200 Sean (masked) visits LT19 at time 0.230 Yong Qi (masked) leaves LT19 at time 0.300 Yong Qi (masked) met Eric (masked) at time 0.330 Eric (masked) met De Zhang (masked) at time 0.340 Prof Terence (masked) leaves LT19 at time 0.400 Prof Henry (masked) visits i3 at time 0.450 Initial Disease Carrier (masked) leaves LT19 at time 0.500 Kevin (masked) leaves LT19 at time 0.500 Sean (masked) leaves LT19 at time 0.500 Yong Qi (masked) tests positive for SARS-CoV-2 at time 1.000 De Zhang (masked) visits i3 at time 1.100 Sean (masked) visits i3 at time 1.200 Initial Disease Carrier (masked) visits i3 at time 1.300 De Zhang (masked) leaves i3 at time 1.300 Initial Disease Carrier (masked) leaves i3 at time 1.300 Sean (masked) leaves i3 at time 1.300 Prof Henry (masked) leaves i3 at time 1.300 Prof Henry (masked) met De Zhang (masked) at time 1.300 Eric (masked) met Prof Henry (masked) at time 1.400 De Zhang (masked) tests positive for SARS-CoV-2 at time 3.000 Initial Disease Carrier (masked) visits COM1-B113 at time 3.100 Marcus (masked) visits COM1-B113 at time 3.100 Jerryl (masked) visits COM1-B113 at time 3.200 Yong Qi (masked) visits COM1-B113 at time 3.300 Prof Henry (masked) visits COM1-B113 at time 3.400 Kevin (masked) visits COM1-B114 at time 3.500 Destinee (masked) visits COM1-B114 at time 3.600 Jerryl (masked) tests positive for SARS-CoV-2 at time 3.600 Yong Qi (masked) leaves COM1-B113 at time 3.600 Initial Disease Carrier (masked) leaves COM1-B113 at time 4.000 Marcus (masked) leaves COM1-B113 at time 4.000 Jerryl (masked) leaves COM1-B113 at time 4.000 Prof Henry (masked) leaves COM1-B113 at time 4.000 Kevin (masked) leaves COM1-B114 at time 4.000 Destinee (masked) leaves COM1-B114 at time 4.000 Siddarth Raj (masked) met Yong Qi (masked) at time 5.000 Xuan Ming (masked) met Yong Qi (masked) at time 7.000 Xuan Ming (masked) visits ION Orchard at time 9.000 Jerryl (masked) visits ION Orchard at time 9.400 Xuan Ming (masked) leaves ION Orchard at time 9.500 Xuan Ming (masked) met Le Yang (masked) at time 10.000 Le Yang (masked) met Joel (masked) at time 10.500 Mario (masked) met Jeremy (masked) at time 10.500 Bryan (masked) visits ION Orchard at time 10.500 Geyu (masked) visits ION Orchard at time 10.500 Jerryl (masked) leaves ION Orchard at time 10.500 Geyu (masked) leaves ION Orchard at time 10.500 Bryan (masked) leaves ION Orchard at time 10.500 Mario (masked) test negative for SARS-CoV-2 at time 11.000 ===== STATISTICS ===== Infected population: 7 Total Population: 19 ===== SIMULATION COMPLETED =====  ===== RUNNING SIMULATION ===== Mask policy implemented and SHN issued  Initial Disease Carrier (masked) visits LT19 at time 0.000 Prof Henry (masked) visits LT19 at time 0.000 Prof Terence (masked) visits LT19 at time 0.000 Yong Qi (masked) visits LT19 at time 0.100 Kevin (masked) visits LT19 at time 0.110 Prof Henry (masked) leaves LT19 at time 0.200 Sean (masked) visits LT19 at time 0.230 Yong Qi (masked) leaves LT19 at time 0.300 Yong Qi (masked) met Eric (masked) at time 0.330 Eric (masked) met De Zhang (masked) at time 0.340 Prof Terence (masked) leaves LT19 at time 0.400 Prof Henry (masked) visits i3 at time 0.450 Initial Disease Carrier (masked) leaves LT19 at time 0.500 Kevin (masked) leaves LT19 at time 0.500 Sean (masked) leaves LT19 at time 0.500 Yong Qi (masked) tests positive for SARS-CoV-2 at time 1.000 Initial Disease Carrier (masked) has been served a SHN that ends at 14.100 Prof Henry (masked) has been served a SHN that ends at 14.100 Prof Terence (masked) has been served a SHN that ends at 14.100 Kevin (masked) has been served a SHN that ends at 14.110 Sean (masked) has been served a SHN that ends at 14.230 Eric (masked) has been served a SHN that ends at 14.330 De Zhang (masked) visits i3 at time 1.100 De Zhang (masked) leaves i3 at time 1.300 De Zhang (masked) tests positive for SARS-CoV-2 at time 3.000 Eric (masked) has been served a SHN that ends at 14.340 Prof Henry (masked) has been served a SHN that ends at 15.100 Marcus (masked) visits COM1-B113 at time 3.100 Jerryl (masked) visits COM1-B113 at time 3.200 Destinee (masked) visits COM1-B114 at time 3.600 Jerryl (masked) test negative for SARS-CoV-2 at time 3.600 Marcus (masked) leaves COM1-B113 at time 4.000 Jerryl (masked) leaves COM1-B113 at time 4.000 Destinee (masked) leaves COM1-B114 at time 4.000 Xuan Ming (masked) visits ION Orchard at time 9.000 Jerryl (masked) visits ION Orchard at time 9.400 Xuan Ming (masked) leaves ION Orchard at time 9.500 Xuan Ming (masked) met Le Yang (masked) at time 10.000 Le Yang (masked) met Joel (masked) at time 10.500 Mario (masked) met Jeremy (masked) at time 10.500 Bryan (masked) visits ION Orchard at time 10.500 Geyu (masked) visits ION Orchard at time 10.500 Jerryl (masked) leaves ION Orchard at time 10.500 Geyu (masked) leaves ION Orchard at time 10.500 Bryan (masked) leaves ION Orchard at time 10.500 Mario (masked) test negative for SARS-CoV-2 at time 11.000 ===== STATISTICS ===== Infected population: 4 Total Population: 19 ===== SIMULATION COMPLETED ===== | | --- | |