CSE443 - Object Oriented Analysis and Design Final Report

Gökçe Nur Erer - 171044079

1 Introduction

This simulation visualizes a currently active epidemic in a society. When simulation starts there is a single infected person and the simulation aims to visualize all the infections, collisions, hospitalizations and deaths in this epidemic. Before going into the details of the implementation of this simulation it is important to explain how the simulation works. Simulation itself gets updated with 50ms intervals, and simulation itself updates every other thing inside of it with this time interval. Every object who needs an update during the simulation has a updating method which works with this 50ms time interval.

2 Implementation Details

2.1 Human Creation System

In this simulation, human creation is done with the Factory Design Pattern. The reason that the factory pattern is used for this cause is that human creation involves lots of different randomizing variables and also human needs to get a lot of parameters to be constructed. To minimize the complexity of the human creation for the clients and also to be able to create human groups this design pattern is chosen.

HumanFactory class can either create a human, or a human group with a given size. Every human or human group is totally randomized in every parameter.

Human and the view of the Human are seperated due to the fact that the view should only deal with the drawing of the human, and shouldn't care about the other events that happen to the human itself. For this HumanViewModel class is created and it will be explained more in the **GUI Details** section of this report.

2.2 Human Collision Detection System

In this simulation the human collision detection is implemented by the mediator design pattern. The reason of using this pattern is that humans shouldn't know about each other when they collide with another.

To implement this pattern a class called CollisionMediator is created. CollisionMediator class both checks if a collision happened in that time frame in the simulation and also deals with the collisions that happened. Simulation calls CollisionMediator's checkCollisions method for everytime it updates itself.

2.2.1 Collision Detection

The collision detection is done by checking two human's positions. (P.S.Position is represented as the top left corner of every view) If their position are on the same location they are considered as collided. Noone else can enter to this collision but another collision can happen on the same location. What is meant by that is that if two people collides in a location another two people can collide in the same location at the same time but they are considered two different collisions.

2.2.2 Collision Handling

The collision handling makes sure that the infection process only applies if a person is infected and the other one is not infected. The probability of the infection is calculated if only one of the people in the collision is infected. After finding the probability a random number is selected between 1 and 100. If the random number is smaller than the probability times 100 that meant that the person should be infected and else they shouldn't. After setting a human infected, human's countdowns for death and hospitalization are started by using human's methods.

2.3 Plotting System

The plotting in this simulation is implemented by a third-party library called XChart. It is chosen since it is clear and understandable in usage and has a perfect compatibility with Java Swing.

In this simulation this plotting is represented with a class called EpidemicGraphWindow. This class holds a XChart chart in it and few GUI frames which will be explained more in **GUI Details** section of this report.

Whenever the simulation gets updated this plot's contents are updated as well. When updating the plot the total death count and the total case count is given to the plot from the simulation.

2.4 Hospitalization System

In this simulation hospital is thought as a single structure. That is why it is implemented as a singleton. Hospitalization system is multithreaded. When a human needs hospitalization it creates a thread and tries to acquire a ventilator in the hospital. Everything is synchronized by using synchronized keyword and wait, notifyAll mechanisms. To be more detailed about the methods of this system if 25 seconds passed after the initial infection of a human, it tries gets hospitalized by opening a thread (only if there is an available ventilator, which means current ventilator count is not 0). If it can it spends 10 seconds then uses the discharge method to release a ventilator hence increases the ventilator count and notifies other threads who are waiting for ventilators. If it can't get hospitalized it stays infected and keeps on moving around the society while its hospitalization thread waits until another hospitalization thread notifies it.

P.S In the GUI sometimes it may be seem like the hospitalization count is always maxed. But the reason why it stays like that is that the threads act quicker than the update itself and fills the emptying slot almost immediately.

2.5 Death System

In this simulation a human dies if 100 * (1 - Z) seconds pass after its initial infection if it doesn't get hospitalized. When a human dies its both view and itself gets removed from simulation by removing population and population's view.

2.6 Class Implementation Details

2.6.1 Human Class

Before going into the details of the human class two enums created for this class should be explained further:

MovementDirection Enum: This enum defines the movement direction of a human. It can be EAST, WEST, NORTH, SOUTH, NORTHEAST, NORTHWEST, SOUTHEAST, SOUTHWEST.

HealthState Enum: This enum defines the health state of a human. It can be Healthy, Infected, Hospitalized or Dead. This enum also kind of helps implementing a state pattern of a humans health condition.

The methods implemented in the Human class are as follows (besides setters and getters):

stall(int stallDuration): This method sets a human's stall time to 0 and sets its stalling duration. Also it disables human from moving by setting a variable called canMove to false. This method is called when a collision happens.

onTick(int deltaTime): This method updates a human at a given time interval. This method represents basically everything human needs to check to update itself. It checks the following:
- If it can move or not and depending on the result it updates the position of the human.
- If its infected and needs hospitalization or if the human is should die depending on the results, if it needs hospitalization it creates a thread to be accepted to the hospital, if human needs to die it is set as dead. - If its hospitalized and the hospitalization time is done, it calls the discharge method in the hospital to leave the hospital. - If the human can't move and the stall duration is over, it removes the stall status and randomizes a new location to locate the human.

updatePos(boolean forceChangeDirection): This method updates the position of the human depending on the speed and location of the human. If the boundaries of the simulation area exceeds the movement direction is randomly chosen and it leads the human to move in a different direction. The forceChangeDirection parameter in this method is used to change direction even though human didn't exceed the boundary limits. This parameter is setted true when the human is out of the hospital.

2.6.2 Collision Mediator Class

This class detects the collisions in a population and handles the collisions. For this these methods are implemented (besides setter and getters):

checkCollisions(): This method checks if any human couple have collided in a simulation and handles the collision if so.

handleCollision(Human human1, Human human2): This method gets two human who are collided and calculates an infection probability and depending on this probability one person gets infected or not. The details are explained in the **Collision Handling** section above.

2.6.3 EpidemicSimulator Class

This class is the base class for the simulation it handles the main GUI and the updating of every item in the simulation and its GUI. To do that it implements the following methods (besides getter and setters):

createWindow(): This method creates the overall window with the initial pop ups.

createSimulationPanel(): This method creates the area that the human views will move and collide.

createStatisticsPanel(): This method creates the bottom part of the simulation window which has the statistics and buttons. It uses createControlPanel method to add the buttons.

createControlPanel(): This method creates the button panel.

startSimulation(): This method starts the simulation, if the simulation didn't start before, it creates a population with the given count at the initial pop up, it infects a random human in the population, initializes the hospital with given population size, initializes the human views and starts a timer that will call a custom action listener called EpidemicActionListener with 50ms.

onTick(): This method updates the simulation. It updates humans by using updateHumans() method, checks collisions by using the collision mediator, updates human views by calling their onTick() method, updates plot if a second passed, updates count labels by using updateLabels() method and increases the timer.

infectRandomHuman(): This method infects a random human in the population.

updateHumans(): This method updates the humans by using their onTick() method. Also certain checks are made in this method to increase total hospitalized count and remove dead people from the views and from the overall simulation.

updateLabels(): This method updates the labels with the updated values of the counts hold in the simulation.

EpidemicActionListener Inner Class: This inner class is a custom ActionListener which calls onTick() method of the EpidemicSimulator class when 50ms passes. The play and pause mechanism actually relies on this class. When the play button is pressed a variable in the simulator called isRunning is setted true. When pause button is pressed it is setted false. EpidemicActionListener checks if the variable is true or false. It only updates the simulator hence calls the onTick() method of the simulator if the variable is true.

2.6.4 Hospital Class

This class represents a hospital and it implements singleton pattern. It supports multithreading. Following methods are implemented to enable the hospitalization function:

getInstance(): This method is created to implement the singleton pattern. It creates an hospital if its not created before, else it returns the instance that was created already.

initialize(int maxVentilatorCount): This method initializes the ventilator count in a hospital.

accept(Human human): This method lets a human get a ventilator in a hospital if there is enough ventilators. If not, this method makes that thread wait until it is notified by the discharge method called by another thread. If a human can get a ventilator its health state is setted to be hospitalized.

discharge(Human human): This method discharges a human from the hospital and notifies all waiting threads who are trying to get a ventilator from the hospital by using accept method. When discharging a human from the hospital, its health state is setted healthy and its position and movement direction is randomized again.

2.6.5 Population Class

This class is an abstraction of a list that holds humans. It implements to following methods (besides getters and setters):

addHuman(Human human): This method adds a human to the population.

addHumanGroup(List¡Human¿ humanGroup): This method adds a human group to the population.

removeHuman(Human human): This method removes the given human from the population.

infectRandomHuman(): This method infects a random human in the population.

getMaskUsagePercentage(): This method returns the mask usage percentage in the population.

getAvgSocialDistance(): This method returns the average social distancing practiced in the population.

getHealthyCount(): This method returns the count of the humans that are currently healthy in the population.

getHospitalizedCount(): This method returns the count of the humans that are currently hospitalized in the population.

getInfectedCount(): This method returns the count of the humans that are currently infected in the population.

3 GUI Details

3.1 HumanViewModel Class

In this simulation human and its view are seperated and they are different classes. HumanViewModel only deals with the drawing of the human and doesn't do anything else. It extends from JComponent and it overrides the paintComponent method to draw a 5x5 square, which is red if the human is infected and green if the human is healthy. It implements this methods (besides the getters and setters):

paintComponent(Graphics g): This method overrides the paintComponent method from the JComponent class and it draws a 5x5 square, which is red if the human who is represented is infected and green if the human is healthy.

onTick(): This method repaints the square when its called since the human that it represents is moving. This method is called in the simulation when it updates itself. This method also checks if a view should be shown or not depending on the human that is represented with the view is hospitalized or not.

3.2 PopulationViewModelClass

The purpose of this class is to bring an abstraction to the population's view. This class is only a wrapper for a HumanViewModel list. It implements these methods (besides getters and setters):

getWithHuman(Human human) This methods returns a HumanViewModel which represents the requested human in a PopulationViewModel.

updatePopulation(Population population) This method updates the population that this view represents.

3.3 EpidemicGraphWindow Class

This class represents a graph window in the simulation. This window can be opened with a button at the simulation GUI. It holds a XChart chart inside and implements these methods to create and update a chart:

createChart(): This method initializes the XChart chart in this class.

createPlotSeries(String name): This method adds a plot series to the chart with a given name.

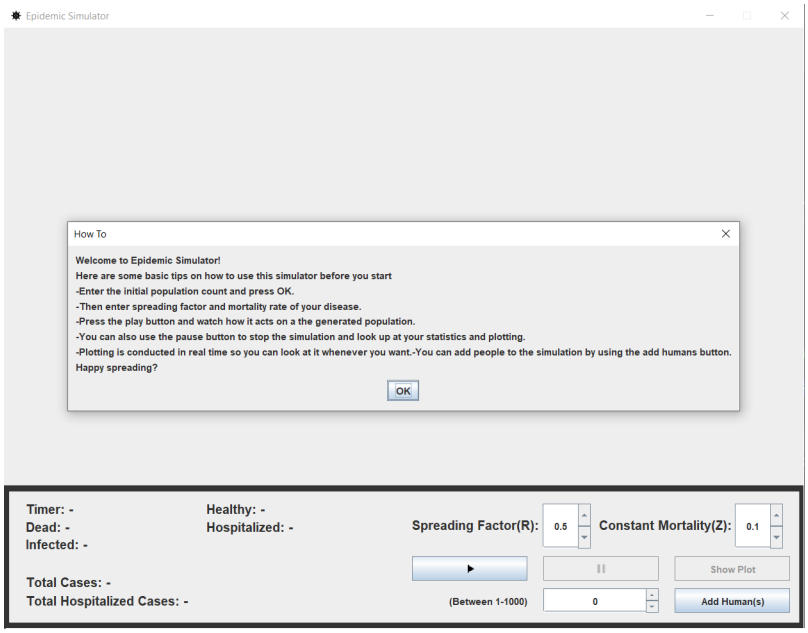
openPlot(): This method shows the plot.

setPopulationParameters(double spreadingFactor, double mortalityRate, double maskUsage, double avgSocialDistance): Besides the chart itself this window also shows the population parameters. This method sets those values to be shown.

onTick(): This method updates the values by adding the current values to the series.

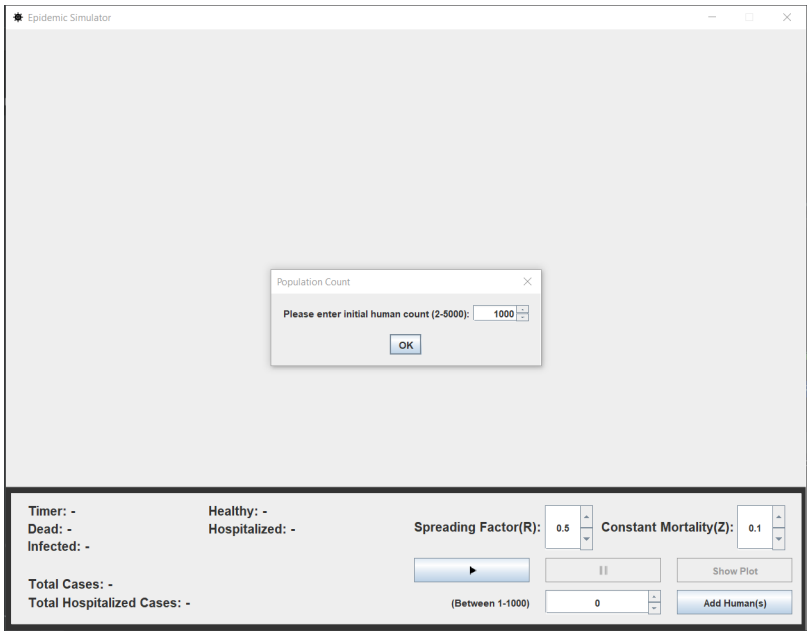
updatePopulationParameters(double spreadingFactor, double mortalityRate, double maskUsage, double avgSocialDistance): Since this window also shows the population parameters when new people gets added to the population, the parameters will change. This method updates the parameters to the newer ones which are given.

3.4 Epidemic Simulation GUI Details and How It Works

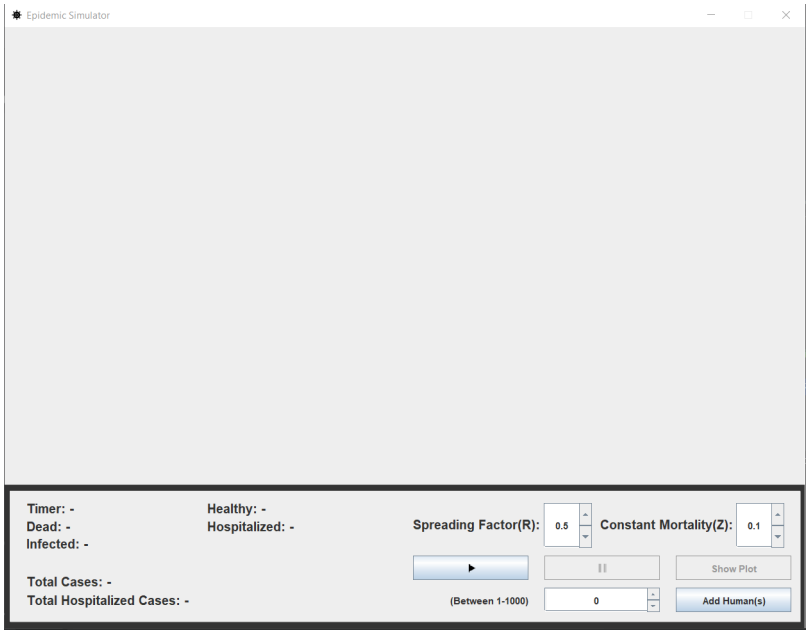


This image shows the opening screen of this application. It gives a brief how-to

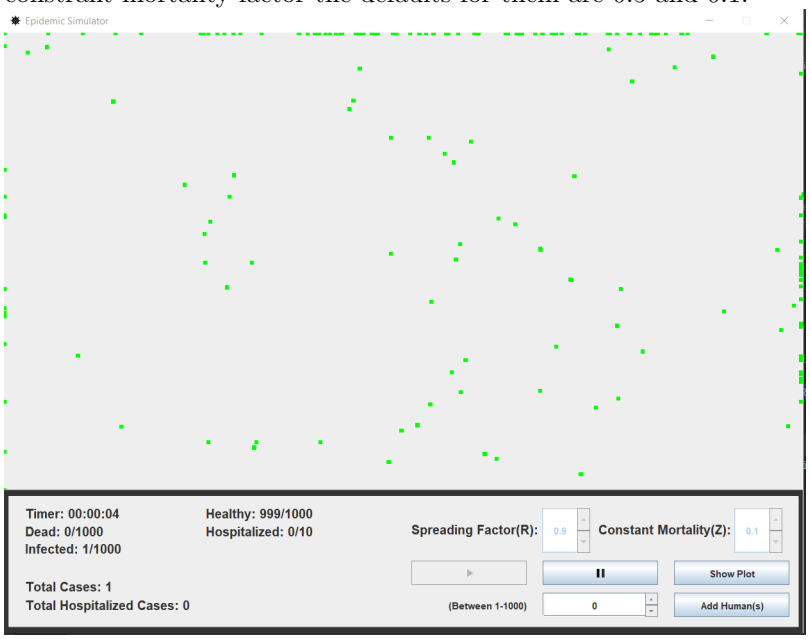
on using the simulation.



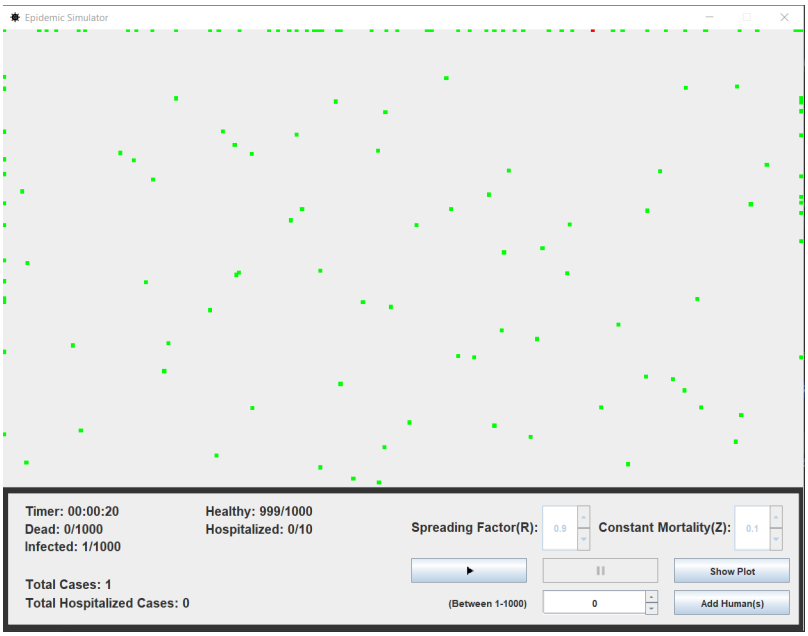
After clicking OK or X at the previous screen this screen is opened. In here the initial population size can be entered. The number must be between 2 and 1000 and the default value is 2.



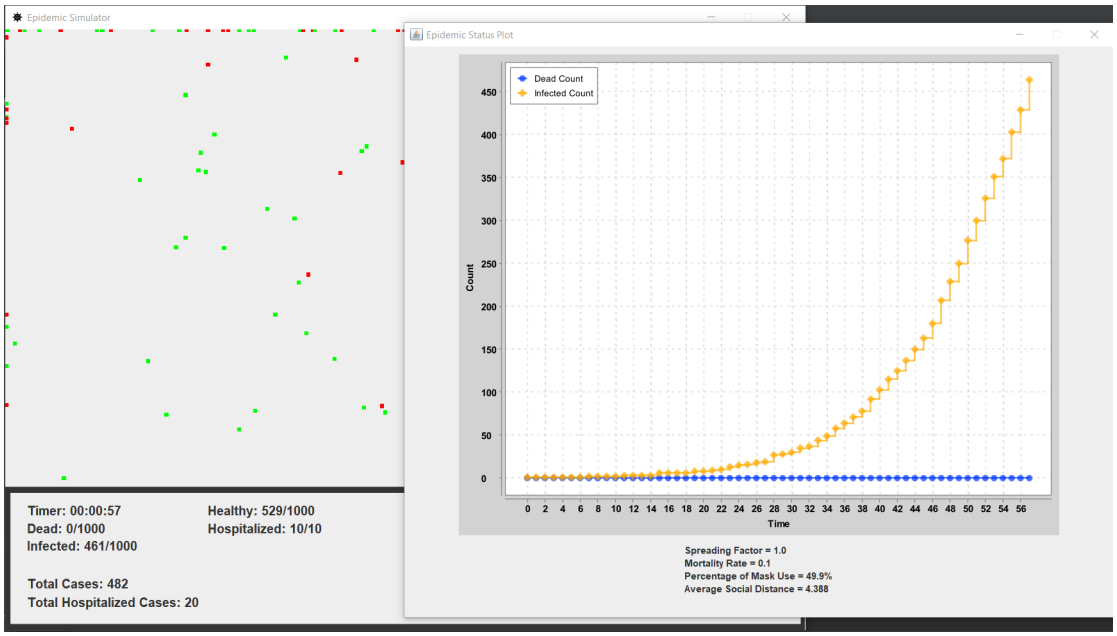
After entering the initial population size. This screen opens up and by setting up spreading factor and constant mortality the simulation can be started by pressing the play button. If nothing is entered to the spreading factor and constrant mortality factor the defaults for them are 0.5 and 0.1.



After starting the simulation the screen updates itself like this image. Now the simulation can be paused, new humans can be added and the plot can be shown. Also the statistic panel is updated and the numbers are initialized and keeps on changing when there is a difference in the simulation. Timer represents the passed time, healthy represents the currently healthy people count in the population, dead represents the dead people in the population, hospitalized represents the current number of people who are hospitalized/ventilator count, infected represents the currently infected count in the population. Total cases represents the overall infection count which means if a person gets infected then heals then gets infected again, this count will be incremented again but this will not be the case at the infected label previously mentioned. Same thing is in the total hospitalized count as well. Every hospitalization is counted in the total hospitalized count but at hospitalized label only the ones that are currently in hospital are shown.

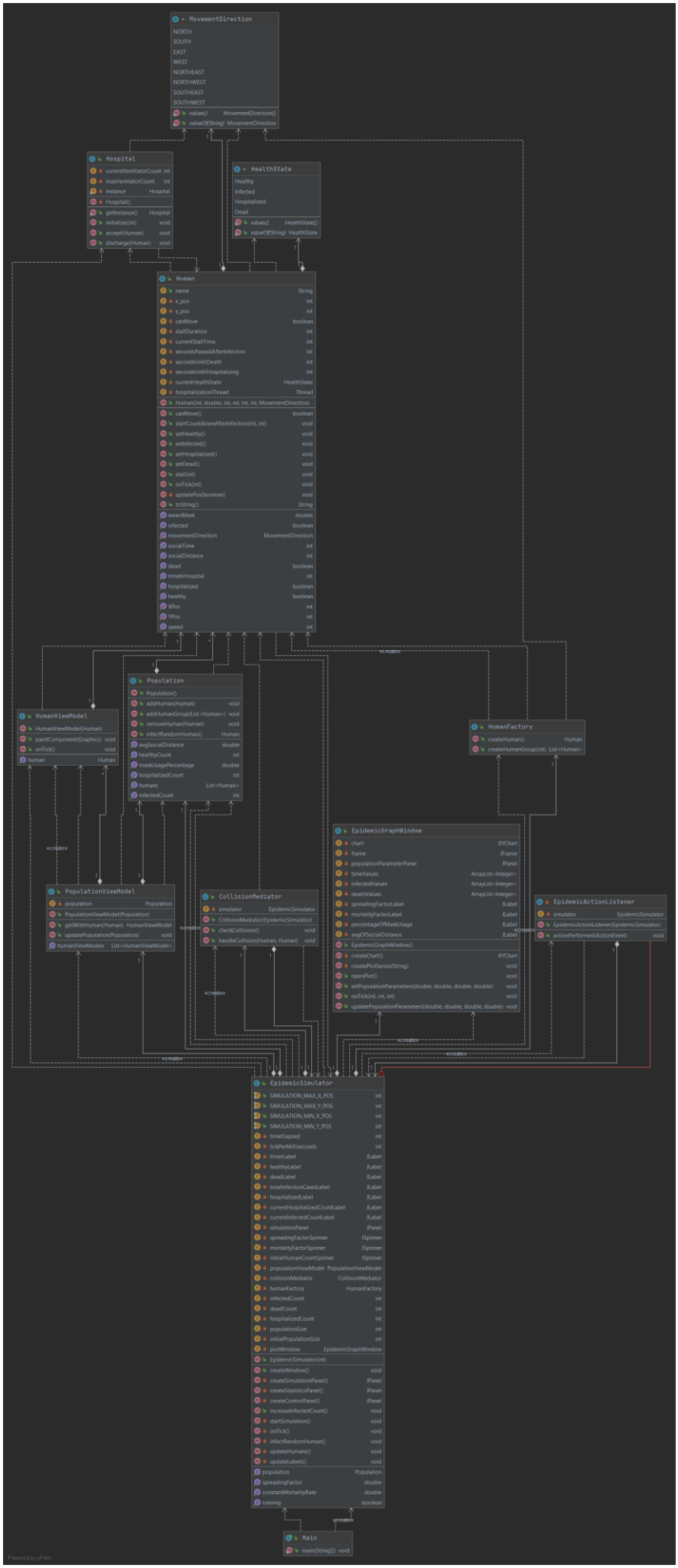


This is the screen when the simulation is paused.



This is the screen when the plotting is opened. The plotting is updated at every second since death and infection is depended on the seconds but not mil-

liseconds. This screen also shows the parameters to the population at the time and it gets updated if new human/human groups are added by using the add human(s) button. The spinner next to that button represents the human count that will be added to the population and it can only be between 1 and 1000 and the default value is 1 if nothing is entered.


MovementDirection
NORTH
SOUTH
EAST
WEST
NORTHEAST
NORTHWEST
SOUTHEAST
SOUTHWEST
values() MovementDirection[]
valueOf(String) MovementDirection
Hospital
currentVentilatorCount int
maxVentilatorCount int
instance Hospital
Hospital()
getInstance() Hospital
initialize(int) void
accept(Human) void
discharge(Human) void
HealthState
Healthy
Infected
Hospitalized
Dead
values() HealthState[]
valueOf(String) HealthState
Human
name String
x_pos int
y_pos int
canMove boolean
stallDuration int
currentStallTime int
secondsPassedAfterInfection int
secondsUntilDeath int
secondsUntilHospitalizing int
currentHealthState HealthState
hospitalizationThread Thread
Human(int, double, int, int, int, int, MovementDirection)
canMove() boolean
startCountdownAfterInfection(int, int) void
setHealthy() void
setInfected() void
setHospitalized() void
setDead() void
stall(int) void
onTick(int) void
updatePos(boolean) void
toString() String
wearsMask double
infected boolean
movementDirection MovementDirection
socialTime int
socialDistance int
dead boolean
timeInHospital int
hospitalized boolean
healthy boolean
XPos int
YPos int
speed int
«create»
Population
Population()
addHuman(Human) void
addHumanGroup(List<Human>) void
removeHuman(Human) void
infectRandomHuman() Human
avgSocialDistance double
healthyCount int
maskUsagePercentage double
hospitalizedCount int
humans List<Human>
infectedCount int
HumanViewModel
HumanViewModel(Human)
paintComponent(Graphics) void
onTick() void
human Human
HumanFactory
createHuman() Human
createHumanGroup(int) List<Human>
«create»
EpidemicGraphWindow
chart XYChart
frame JFrame
populationParameterPanel JPanel
timeValues ArrayList<Integer>
infectedValues ArrayList<Integer>
deathValues ArrayList<Integer>
spreadingFactorLabel JLabel
mortalityFactorLabel JLabel
percentageOfMaskUsage JLabel
avgOfSocialDistance JLabel
EpidemicGraphWindow()
createChart() XYChart
createPlotSeries(String) void
openPlot() void
setPopulationParameters(double, double, double, double) void
onTick(int, int, int) void
updatePopulationParameters(double, double, double, double) void
PopulationViewModel
population Population
PopulationViewModel(Population)
getWithHuman(Human) HumanViewModel
updatePopulation(Population) void
humanViewModels List<HumanViewModel>
CollisionMediator
simulator EpidemicSimulator
CollisionMediator(EpidemicSimulator)
checkCollisions() void
handleCollision(Human, Human) void
EpidemicActionListener
simulator EpidemicSimulator
EpidemicActionListener(EpidemicSimulator)
actionPerformed(ActionEvent) void
«create»
EpidemicSimulator
SIMULATION_MAX_X_POS int
SIMULATION_MAX_Y_POS int
SIMULATION_MIN_X_POS int
SIMULATION_MIN_Y_POS int
timeElapsed int
tickPerMilliseconds int
timerLabel JLabel
healthyLabel JLabel
deadLabel JLabel
totalInfectionCasesLabel JLabel
hospitalizedLabel JLabel
currentHospitalizedCountLabel JLabel
currentInfectedCountLabel JLabel
simulationPanel JPanel
spreadingFactorSpinner JSpinner
mortalityFactorSpinner JSpinner
initialHumanCountSpinner JSpinner
populationViewModel PopulationViewModel
collisionMediator CollisionMediator
humanFactory HumanFactory
infectedCount int
deadCount int
hospitalizedCount int
populationSize int
initialPopulationSize int
plotWindow EpidemicGraphWindow
EpidemicSimulator(int)
createWindow() void
createSimulationPanel() JPanel
createStatisticsPanel() JPanel
createControlPanel() JPanel
increaseInfectedCount() void
startSimulation() void
onTick() void
infectRandomHuman() void
updateHumans() void
updateLabels() void
population Population
spreadingFactor double
constantMortalityRate double
running boolean
«create»
Main
main(String[]) void
Powered by yFiles