Gebze Technical University
Computer Engineering

CSE 443 - 2020 Fall

FINAL PROJECT REPORT

Epidemic Simulator

FATİH KOÇ
141044013

1 Project Summary

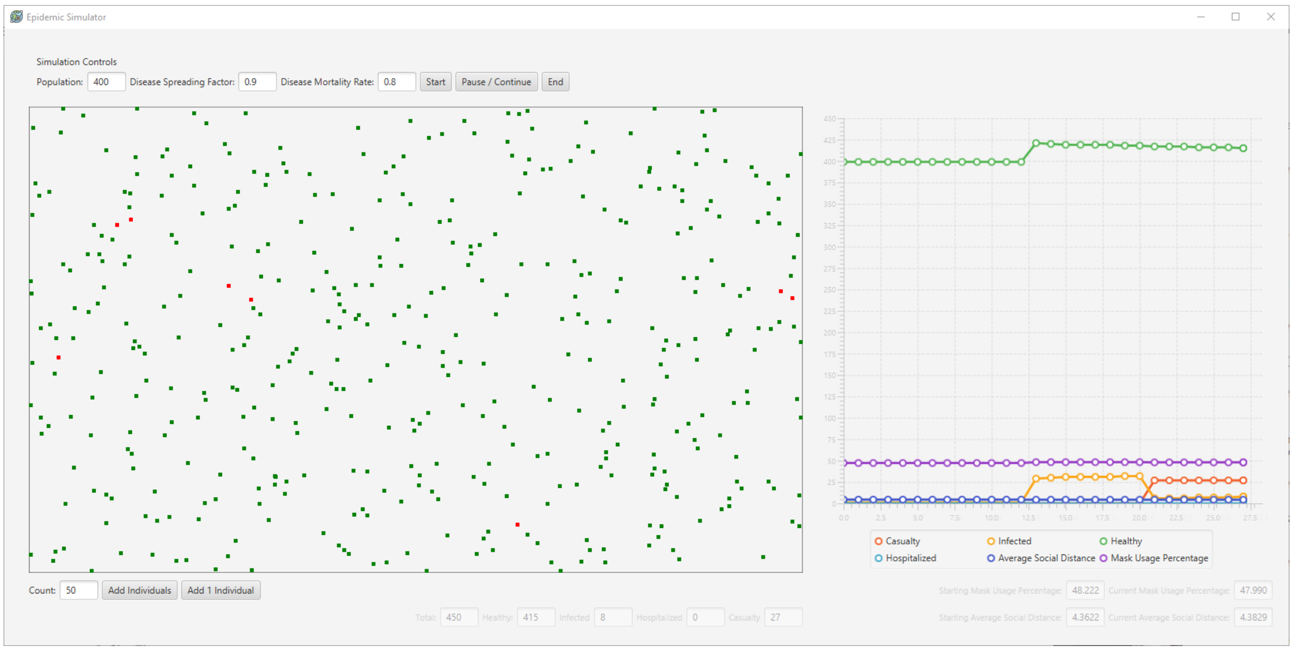
1.1 Problem Definition

Design and implement a visual simulation of an epidemic within a human society by using either Java Swing or FX, with the given details in the shared project file.

1.2 Solution Approach

All requirements from project_v2 implemented in this solution.

To implement user interface, I decided to use JavaFX, which has modern look, easier and cleaner to maintain due to FXML markup files, which helps a lot to write modular, clean and maintainable code.


Epidemic Simulator

Simulation Controls

Population: 400 Disease Spreading Factor: 0.9 Disease Mortality Rate: 0.8

Count: 50

Total: 450 Healthy: 415 Infected 8 Hospitalized 0 Casualty 27

Starting Mask Usage Percentage: 48.222 Current Mask Usage Percentage: 47.990

Starting Average Social Distance: 4.3622 Current Average Social Distance: 4.3829

Figure 1: Program User Interface Layout

Main entities in this solution are **Individual**, **Disease** and **Hospital**. **Mediator** class consists functionalities for these types and its relations between as static methods. **Simulator** class is the main class for our simulation, communicates with **MainController**, all main entities contained in this class and handles state mechanism for simulation. **Common** class contains definitions that is used in multiple places in the program. **fsm** package contains simulation states, **IndividualFactory** generates Individual instances by using **IndividualBuilder** class to generate parameters.

2 Implementation Details

2.1 Simulation Dynamics

2.1.1 Social Distance

In this solution, I assumed that 5px square represents an Individual. To decide where social interaction starts, I assume each person can interact with others in D distance from the borders of the square, like it has an invisible 5+D square around it and can interact when someone else enters it.

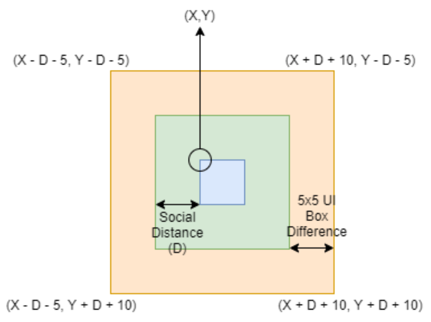


Figure 2: Blue=Individual, Green=Social Contact Distance, To enter a social interaction with blue individual, other individuals coordinates must be inside the orange square

2.1.2 Individual Movement

Individuals can't go out of borders from canvas, can't overlap with each other (except initial positions, due to random selection it can appear). In its direction and updated position, if there is another individual, it should change its direction and try to move again. If there isn't any possible movement direction, individual will wait without moving for this iteration.

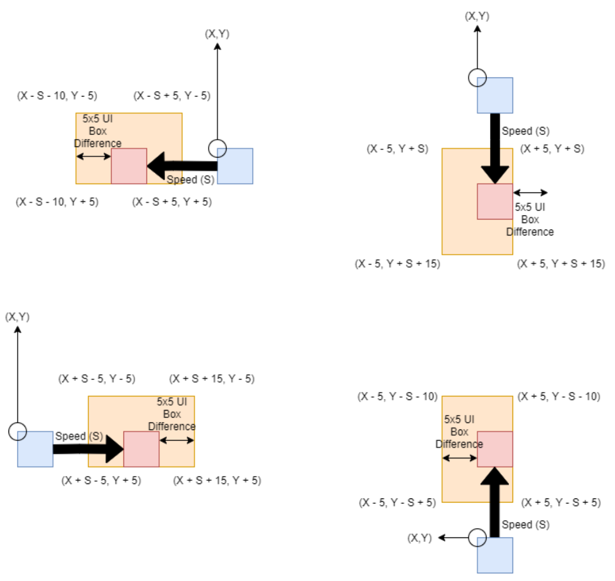


Figure 3: Blue=Individual, Red=Updated position,To move into red position, there should be no Individual in the orange square, or updated position should not be outside of the canvas

2.2 Design Patterns

Design patterns that I used for this solution are:

2.2.1 MVC

The way JavaFX works is very similar to MVC pattern, it uses .fxml files as View, controller classes as Controller in this pattern, to handle UI functions and override events. Model is our Simulation in this scenario. In main class, ActionTimer reloads interface every second, with the method calls from controller which collects information from simulator class.

2.2.2 Game Loop

Simulation should run without any breaks; it should process user input without blocking and update the game state. Loop works in a new thread that processes current state in each iteration, it continues to run if the simulation in initialization or process state. When simulation state changed to another, it stops to run.

2.2.3 Factory

To create Individual class instances, I implemented IndividualFactory, which creates an interface that is easy to use for object creation. With this design pattern, I achieved to avoid tight coupling between the creator and the concrete products and managed to apply Open/Closed Principle by making available to introduce new types of products into the program without breaking existing code.

2.2.4 Builder

With a combination with factory, Builder pattern helped a lot to reduce maintenance cost of the code and applied Single Responsibility Principle much better. In my design, each factory created with a builder object, uses this builder to get each parameter of Individual constructor. This made IndividualFactory is abstracted from the details of parameter generation and take this responsibility into builder object. Right now, there is only RandomIndividualBuilder class implemented which generates every parameter randomly. In future, there can be different algorithms to generate parameters for individuals, for such a scenario, only new builder should be implemented and existing factory will continue to work with it.

2.2.5 Observer (Pub/Sub)

To implement Hospital functionality, Observer pattern helped a lot. Each Individual implements Subscriber interface, Hospital acts as a Publisher. In each iteration of the simulation, Hospital checks every subscribed patients, discharges them if they meet the conditions. Same as this, in each iteration, Simulator checks each Individual, hospitalizes infected ones with respecting the rules of the simulation.

2.2.6 State

Simulation designed as a Finite State Machine with 4 steps: Initialization, Process, Pause and End. State design pattern implements this FSM and flow of state controlled by Simulator class. Each state implements SimulationState interface, which contains execute() method, handles all the work needs to be done in the states.

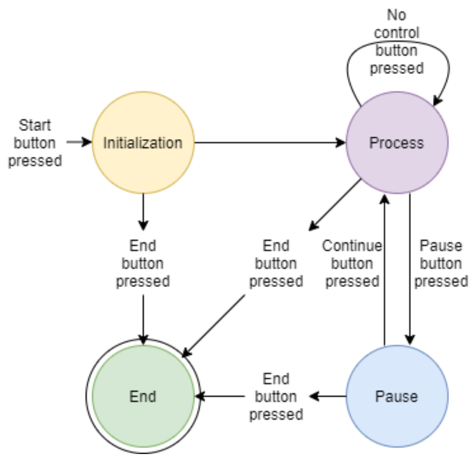
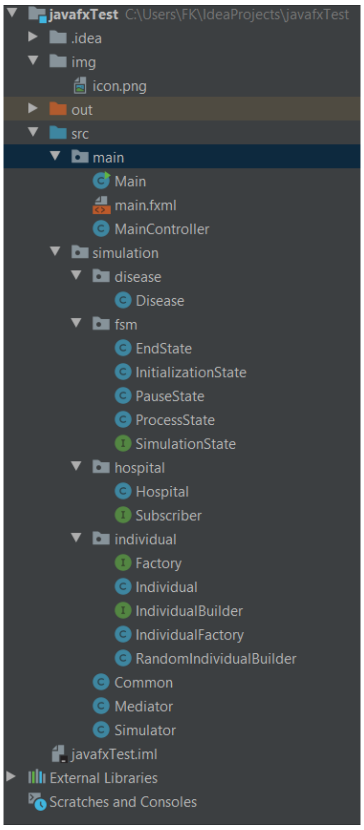


Figure 4: State Machine of simulation

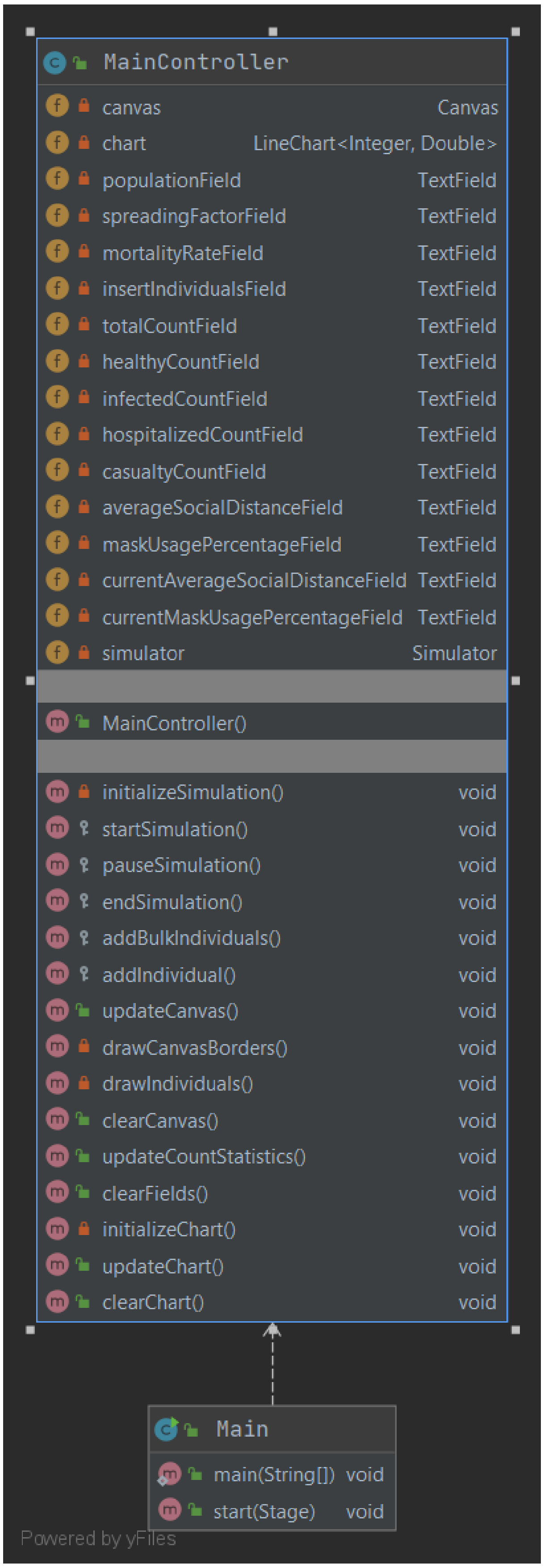
2.3 Class Diagrams

2.3.1 Project Package and File Structure

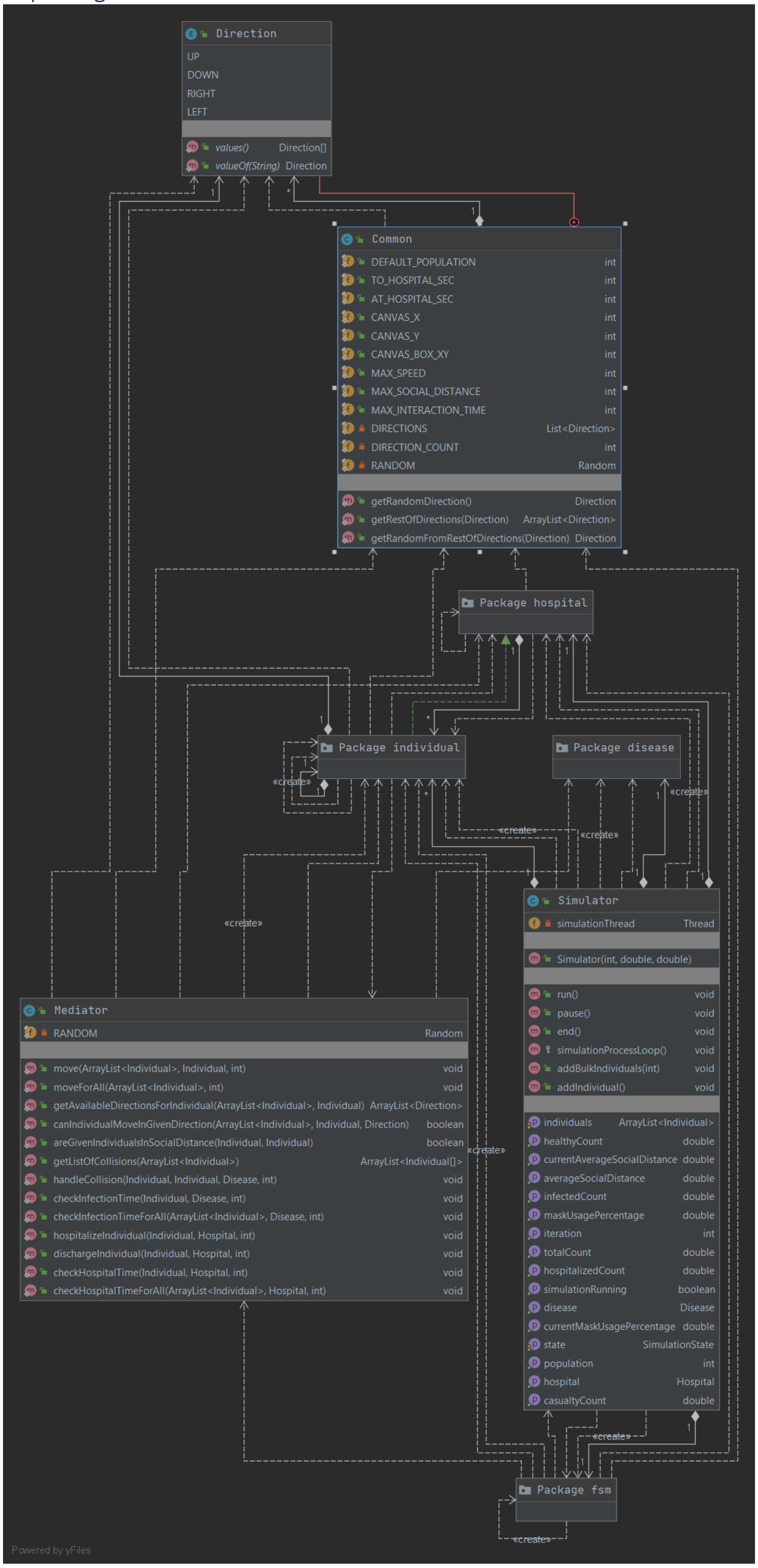


```
javafxTest C:\Users\FK\IdeaProjects\javafxTest
  .idea
  img
    icon.png
  out
  src
    main
      Main
      main.fxml
      MainController
    simulation
      disease
        Disease
      fsm
        EndState
        InitializationState
        PauseState
        ProcessState
        SimulationState
      hospital
        Hospital
        Subscriber
      individual
        Factory
        Individual
        IndividualBuilder
        IndividualFactory
        RandomIndividualBuilder
      Common
      Mediator
      Simulator
  javafxTest.iml
External Libraries
Scratches and Consoles
```

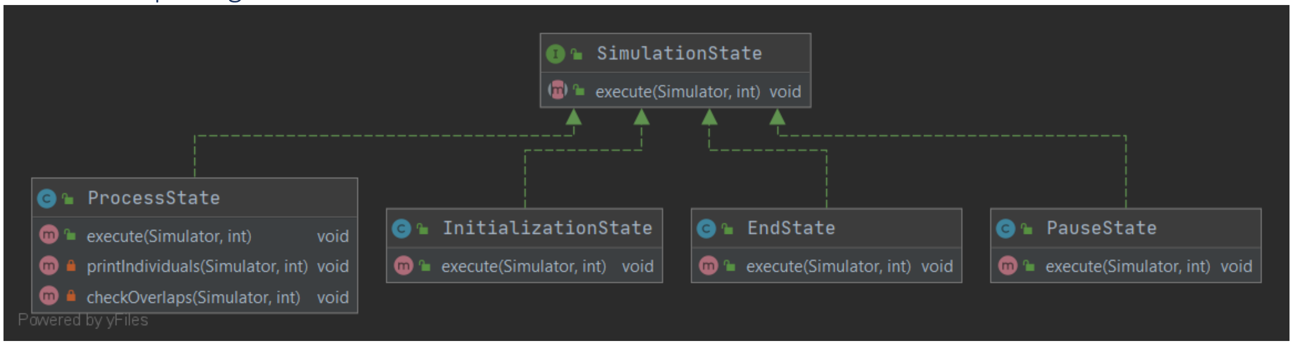
2.3.2 main package


MainController
canvas Canvas
chart LineChart<Integer, Double>
populationField TextField
spreadingFactorField TextField
mortalityRateField TextField
insertIndividualsField TextField
totalCountField TextField
healthyCountField TextField
infectedCountField TextField
hospitalizedCountField TextField
casualtyCountField TextField
averageSocialDistanceField TextField
maskUsagePercentageField TextField
currentAverageSocialDistanceField TextField
currentMaskUsagePercentageField TextField
simulator Simulator
MainController()
initializeSimulation() void
startSimulation() void
pauseSimulation() void
endSimulation() void
addBulkIndividuals() void
addIndividual() void
updateCanvas() void
drawCanvasBorders() void
drawIndividuals() void
clearCanvas() void
updateCountStatistics() void
clearFields() void
initializeChart() void
updateChart() void
clearChart() void
Main
main(String[]) void
start(Stage) void
Powered by yFiles

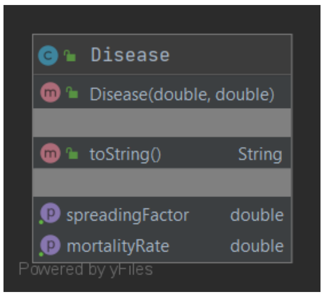
2.3.3 simulation package


Direction
UP
DOWN
RIGHT
LEFT
values() Direction[]
valueOf(String) Direction
Common
DEFAULT_POPULATION int
TO_HOSPITAL_SEC int
AT_HOSPITAL_SEC int
CANVAS_X int
CANVAS_Y int
CANVAS_BOX_XY int
MAX_SPEED int
MAX_SOCIAL_DISTANCE int
MAX_INTERACTION_TIME int
DIRECTIONS List<Direction>
DIRECTION_COUNT int
RANDOM Random
getRandomDirection() Direction
getRestOfDirections(Direction) ArrayList<Direction>
getRandomFromRestOfDirections(Direction) Direction
Package hospital
Package individual
Package disease
«create»
Simulator
simulationThread Thread
Simulator(int, double, double)
run() void
pause() void
end() void
simulationProcessLoop() void
addBulkIndividuals(int) void
addIndividual() void
individuals ArrayList<Individual>
healthyCount double
currentAverageSocialDistance double
averageSocialDistance double
infectedCount double
maskUsagePercentage double
iteration int
totalCount double
hospitalizedCount double
simulationRunning boolean
disease Disease
currentMaskUsagePercentage double
state SimulationState
population int
hospital Hospital
casualtyCount double
Mediator
RANDOM Random
move(ArrayList<Individual>, Individual, int) void
moveForAll(ArrayList<Individual>, int) void
getAvailableDirectionsForIndividual(ArrayList<Individual>, Individual) ArrayList<Direction>
canIndividualMoveInGivenDirection(ArrayList<Individual>, Individual, Direction) boolean
areGivenIndividualsInSocialDistance(Individual, Individual) boolean
getListOfCollisions(ArrayList<Individual>) ArrayList<Individual[]>
handleCollision(Individual, Individual, Disease, int) void
checkInfectionTime(Individual, Disease, int) void
checkInfectionTimeForAll(ArrayList<Individual>, Disease, int) void
hospitalizeIndividual(Individual, Hospital, int) void
dischargeIndividual(Individual, Hospital, int) void
checkHospitalTime(Individual, Hospital, int) void
checkHospitalTimeForAll(ArrayList<Individual>, Hospital, int) void
Package fsm
Powered by yFiles

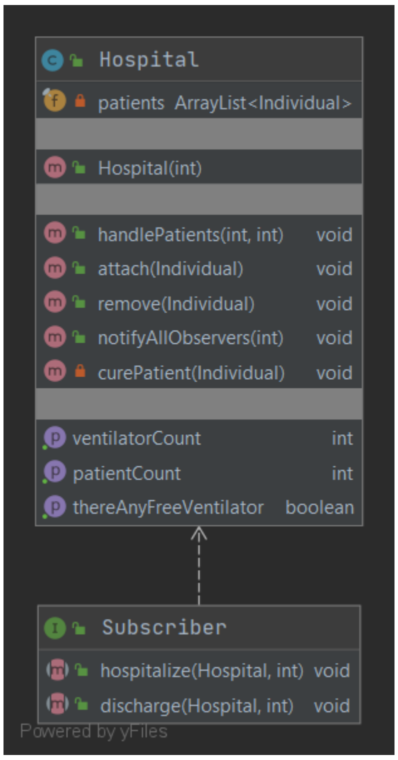
2.3.4 fsm package


SimulationState
execute(Simulator, int) void
ProcessState
execute(Simulator, int) void
printIndividuals(Simulator, int) void
checkOverlaps(Simulator, int) void
InitializationState
execute(Simulator, int) void
EndState
execute(Simulator, int) void
PauseState
execute(Simulator, int) void
Powered by yFiles

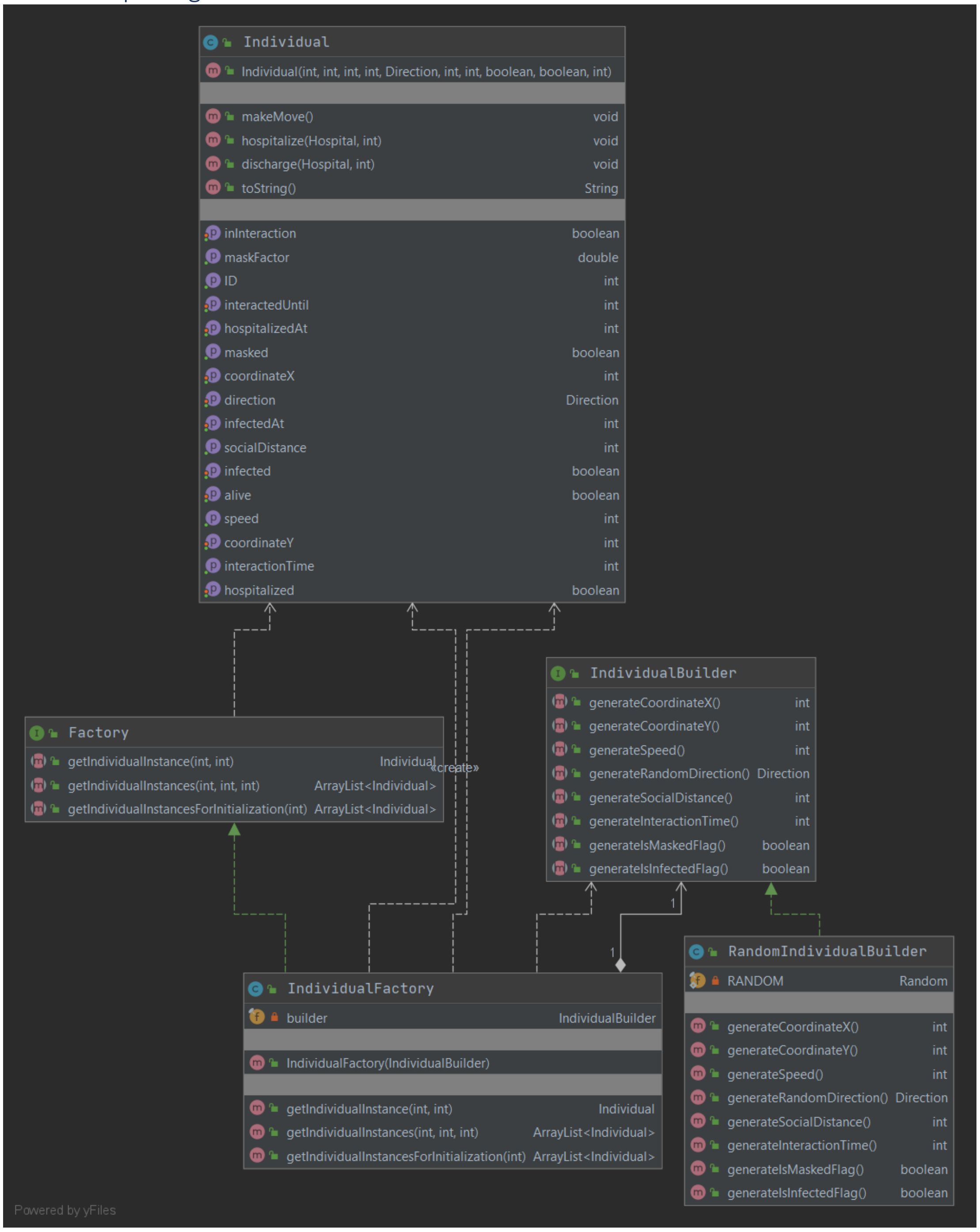
2.3.5 disease package



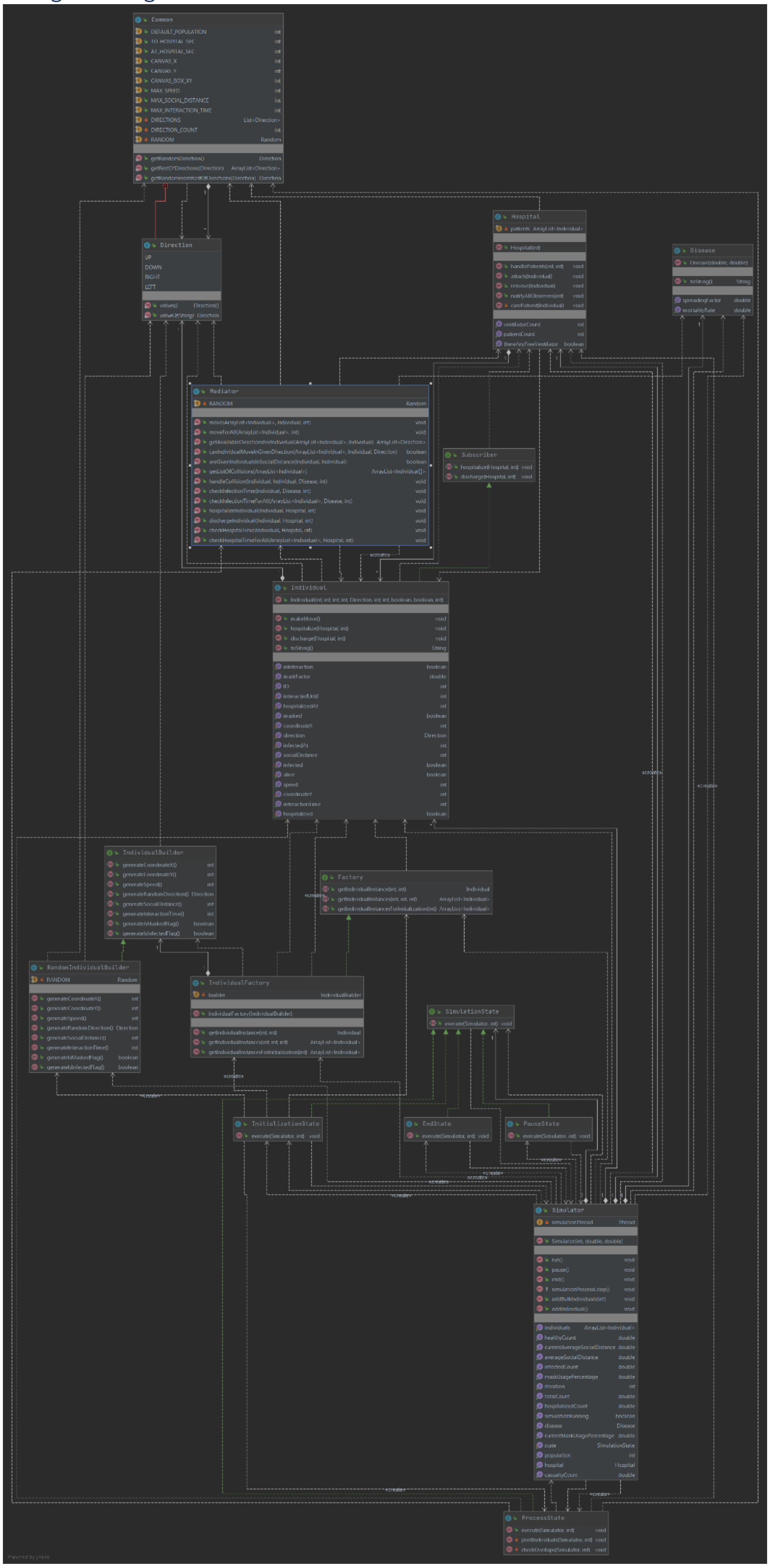
2.3.6 hospital package


Hospital
patients ArrayList<Individual>
Hospital(int)
handlePatients(int, int) void
attach(Individual) void
remove(Individual) void
notifyAllObservers(int) void
curePatient(Individual) void
ventilatorCount int
patientCount int
thereAnyFreeVentilator boolean
Subscriber
hospitalize(Hospital, int) void
discharge(Hospital, int) void
Powered by yFiles

2.3.7 individual package

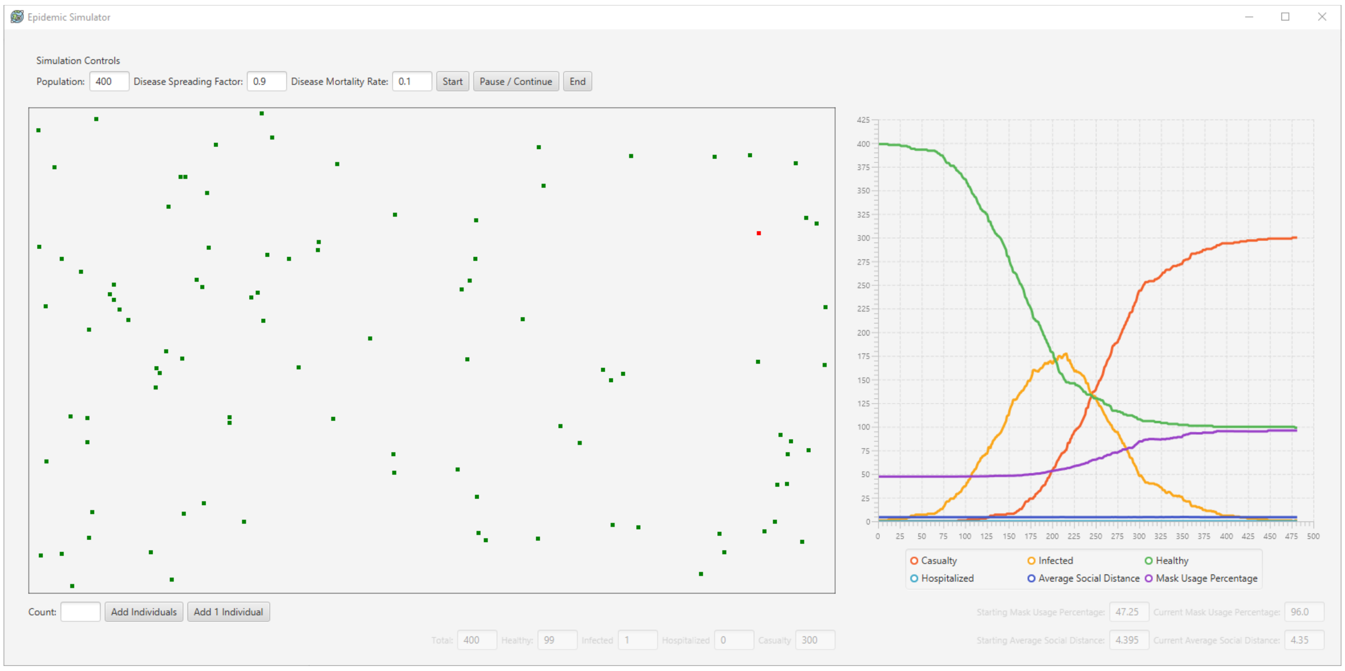

Individual
Individual(int, int, int, int, Direction, int, int, boolean, boolean, int)
makeMove() void
hospitalize(Hospital, int) void
discharge(Hospital, int) void
toString() String
inInteraction boolean
maskFactor double
ID int
interactedUntil int
hospitalizedAt int
masked boolean
coordinateX int
direction Direction
infectedAt int
socialDistance int
infected boolean
alive boolean
speed int
coordinateY int
interactionTime int
hospitalized boolean
IndividualBuilder
generateCoordinateX() int
generateCoordinateY() int
generateSpeed() int
generateRandomDirection() Direction
generateSocialDistance() int
generateInteractionTime() int
generateIsMaskedFlag() boolean
generateIsInfectedFlag() boolean
Factory
getIndividualInstance(int, int) Individual
getIndividualInstances(int, int, int) ArrayList<Individual>
getIndividualInstancesForInitialization(int) ArrayList<Individual>
«create»
1
1
RandomIndividualBuilder
RANDOM Random
generateCoordinateX() int
generateCoordinateY() int
generateSpeed() int
generateRandomDirection() Direction
generateSocialDistance() int
generateInteractionTime() int
generateIsMaskedFlag() boolean
generateIsInfectedFlag() boolean
IndividualFactory
builder IndividualBuilder
IndividualFactory(IndividualBuilder)
getIndividualInstance(int, int) Individual
getIndividualInstances(int, int, int) ArrayList<Individual>
getIndividualInstancesForInitialization(int) ArrayList<Individual>
Powered by yFiles

2.3.8 all class diagrams together


Common
DEFAULT_POPULATION int
TO_HOSPITAL_SEC int
AT_HOSPITAL_SEC int
CANVAS_X int
CANVAS_Y int
CANVAS_BOX_XY int
MAX_SPEED int
MAX_SOCIAL_DISTANCE int
MAX_INTERACTION_TIME int
DIRECTIONS List<Direction>
DIRECTION_COUNT int
RANDOM Random
getRandomDirection() Direction
getRestOfDirections(Direction) ArrayList<Direction>
getRandomFromRestOfDirections(Direction) Direction
Direction
UP
DOWN
RIGHT
LEFT
values() Direction[]
valueOf(String) Direction
Hospital
patients ArrayList<Individual>
Hospital(int)
handlePatients(int, int) void
attach(Individual) void
remove(Individual) void
notifyAllObservers(int) void
curePatient(Individual) void
ventilatorCount int
patientCount int
thereAnyFreeVentilator boolean
Disease
Disease(double, double)
toString() String
spreadingFactor double
mortalityRate double
Mediator
RANDOM Random
move(ArrayList<Individual>, Individual, int) void
moveForAll(ArrayList<Individual>, int) void
getAvailableDirectionsForIndividual(ArrayList<Individual>, Individual) ArrayList<Direction>
canIndividualMoveInGivenDirection(ArrayList<Individual>, Individual, Direction) boolean
areGivenIndividualsInSocialDistance(Individual, Individual) boolean
getListOfCollisions(ArrayList<Individual>) ArrayList<Individual[]>
handleCollision(Individual, Individual, Disease, int) void
checkInfectionTime(Individual, Disease, int) void
checkInfectionTimeForAll(ArrayList<Individual>, Disease, int) void
hospitalizeIndividual(Individual, Hospital, int) void
dischargeIndividual(Individual, Hospital, int) void
checkHospitalTime(Individual, Hospital, int) void
checkHospitalTimeForAll(ArrayList<Individual>, Hospital, int) void
Subscriber
hospitalize(Hospital, int) void
discharge(Hospital, int) void
«create»
Individual
Individual(int, int, int, int, Direction, int, int, boolean, boolean, int)
makeMove() void
hospitalize(Hospital, int) void
discharge(Hospital, int) void
toString() String
inInteraction boolean
maskFactor double
ID int
interactedUntil int
hospitalizedAt int
masked boolean
coordinateX int
direction Direction
infectedAt int
socialDistance int
infected boolean
alive boolean
speed int
coordinateY int
interactionTime int
hospitalized boolean
«create»
«create»
IndividualBuilder
generateCoordinateX() int
generateCoordinateY() int
generateSpeed() int
generateRandomDirection() Direction
generateSocialDistance() int
generateInteractionTime() int
generateIsMaskedFlag() boolean
generateIsInfectedFlag() boolean
Factory
getIndividualInstance(int, int) Individual
getIndividualInstances(int, int, int) ArrayList<Individual>
getIndividualInstancesForInitialization(int) ArrayList<Individual>
«create»
RandomIndividualBuilder
RANDOM Random
generateCoordinateX() int
generateCoordinateY() int
generateSpeed() int
generateRandomDirection() Direction
generateSocialDistance() int
generateInteractionTime() int
generateIsMaskedFlag() boolean
generateIsInfectedFlag() boolean
IndividualFactory
builder IndividualBuilder
IndividualFactory(IndividualBuilder)
getIndividualInstance(int, int) Individual
getIndividualInstances(int, int, int) ArrayList<Individual>
getIndividualInstancesForInitialization(int) ArrayList<Individual>
SimulationState
execute(Simulator, int) void
«create»
«create»
InitializationState
execute(Simulator, int) void
EndState
execute(Simulator, int) void
PauseState
execute(Simulator, int) void
«create»
«create»
«create»
«create»
«create»
Simulator
simulationThread Thread
Simulator(int, double, double)
run() void
pause() void
end() void
simulationProcessLoop() void
addBulkIndividuals(int) void
addIndividual() void
individuals ArrayList<Individual>
healthyCount double
currentAverageSocialDistance double
averageSocialDistance double
infectedCount double
maskUsagePercentage double
iteration int
totalCount double
hospitalizedCount double
simulationRunning boolean
disease Disease
currentMaskUsagePercentage double
state SimulationState
population int
hospital Hospital
casualtyCount double
«create»
«create»
ProcessState
execute(Simulator, int) void
printIndividuals(Simulator, int) void
checkOverlaps(Simulator, int) void
Powered by yFiles

3 Tests, Graphs and Results

3.1 Scenario 1: 400 / 0.9 / 0.1 without Hospital functionality



Population: 400 Disease Spreading Factor: 0.9 Disease Mortality Rate: 0.1

Count:

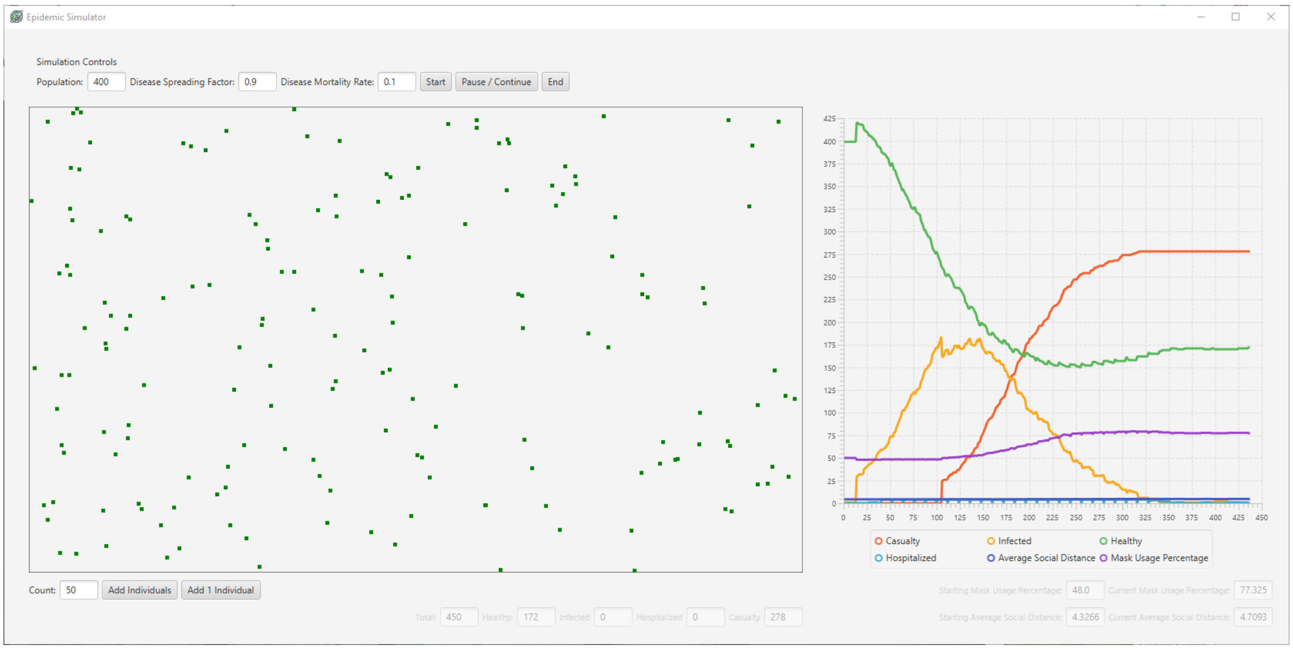
Total: 400 Healthy: 99 Infected 1 Hospitalized 0 Casualty 300

Starting Mask Usage Percentage: 47.25 Current Mask Usage Percentage: 96.0

Starting Average Social Distance: 4.395 Current Average Social Distance: 4.35

In this scenario, first infected individual manage to start an endemic by itself. Without hospital, infected and casualty numbers rise very quickly. Difference in the mask usage percentage between start and end is huge, that can show that casualties are mostly the people without masks, so mask usage has a huge effect on disease spread.

3.2 Scenario 1: 400 / 0.9 / 0.1 with all functionalities



Population: 400 Disease Spreading Factor: 0.9 Disease Mortality Rate: 0.1

Count: 50

Total: 450 Healthy: 172 Infected 0 Hospitalized 0 Casualty 278

Starting Mask Usage Percentage: 48.0 Current Mask Usage Percentage: 77.325

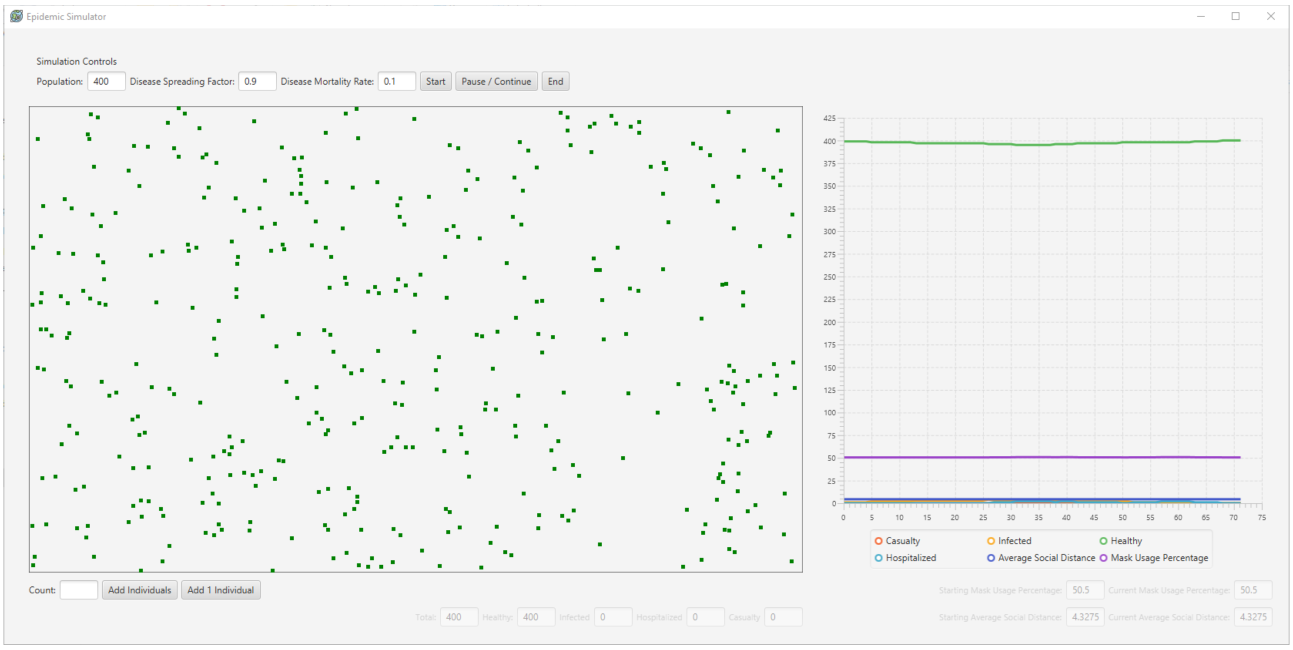
Starting Average Social Distance: 4.3266 Current Average Social Distance: 4.7093

In this scenario, all functions applied.

First observation is, graph change drastically after adding new individuals, its because the first infected person couldn’t manage to spread the infection, probably because of mask factor, with hospital, the infected individual gained its health. To create useful data, random 50 individuals added, almost half of them was infected. It triggered an epidemic.

Second observation, difference in the mask usage percentage between start and end is huge, that can show that casualties are mostly the people without masks, so mask usage has a huge effect on disease spread.

3.3 Scenario 1: 400 / 0.9 / 0.1 with all functionalities


Epidemic Simulator

Simulation Controls

Population: 400 Disease Spreading Factor: 0.9 Disease Mortality Rate: 0.1 Start Pause / Continue End

Count: Add Individuals Add 1 Individual

Total: 400 Healthy: 400 Infected 0 Hospitalized 0 Casualty 0

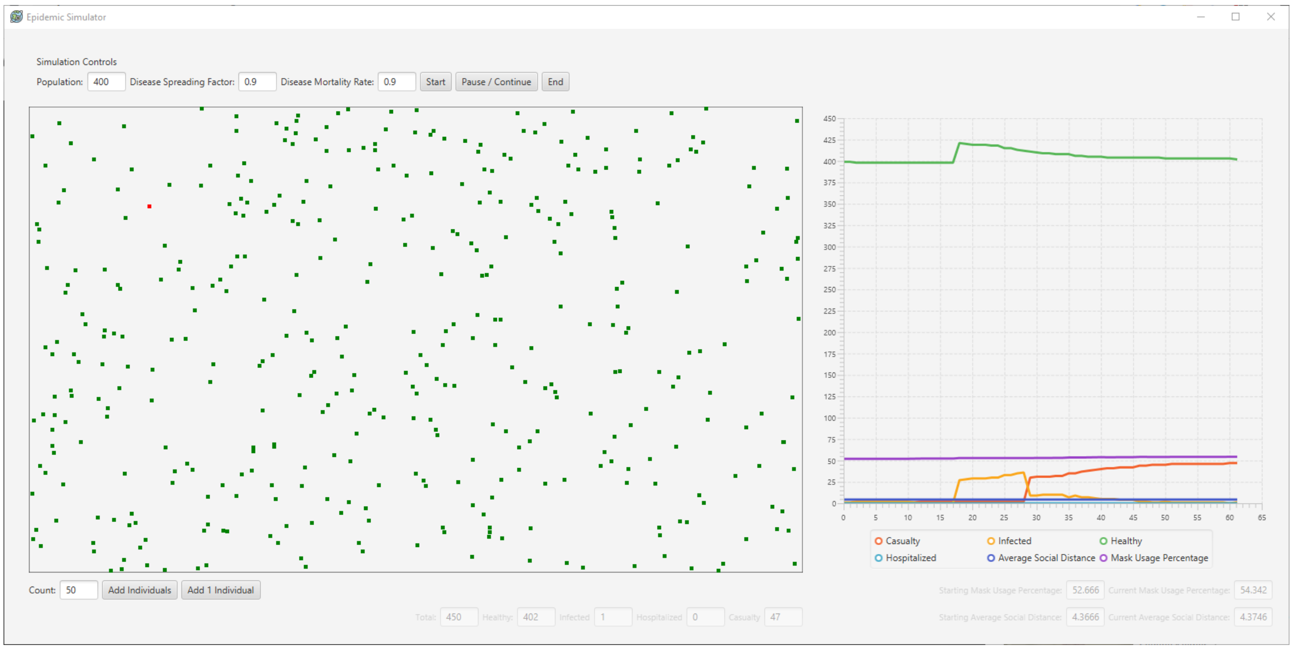
Starting Mask Usage Percentage: 50.5 Current Mask Usage Percentage: 50.5

Starting Average Social Distance: 4.3275 Current Average Social Distance: 4.3275

In this scenario, all functions applied.

First infected person managed to spread the infection to some other individuals, but with hospital, the infected individuals gained its health. Hospital has a huge effect on regulating spreading and mortality, it slows or stops infection spread and pushes casualties much further time or stops it completely.

3.4 Scenario 2: 400 / 0.9 / 0.9 with all functionalities


Epidemic Simulator

Simulation Controls

Population: 400 Disease Spreading Factor: 0.9 Disease Mortality Rate: 0.9 Start Pause / Continue End

Count: 50 Add Individuals Add 1 Individual

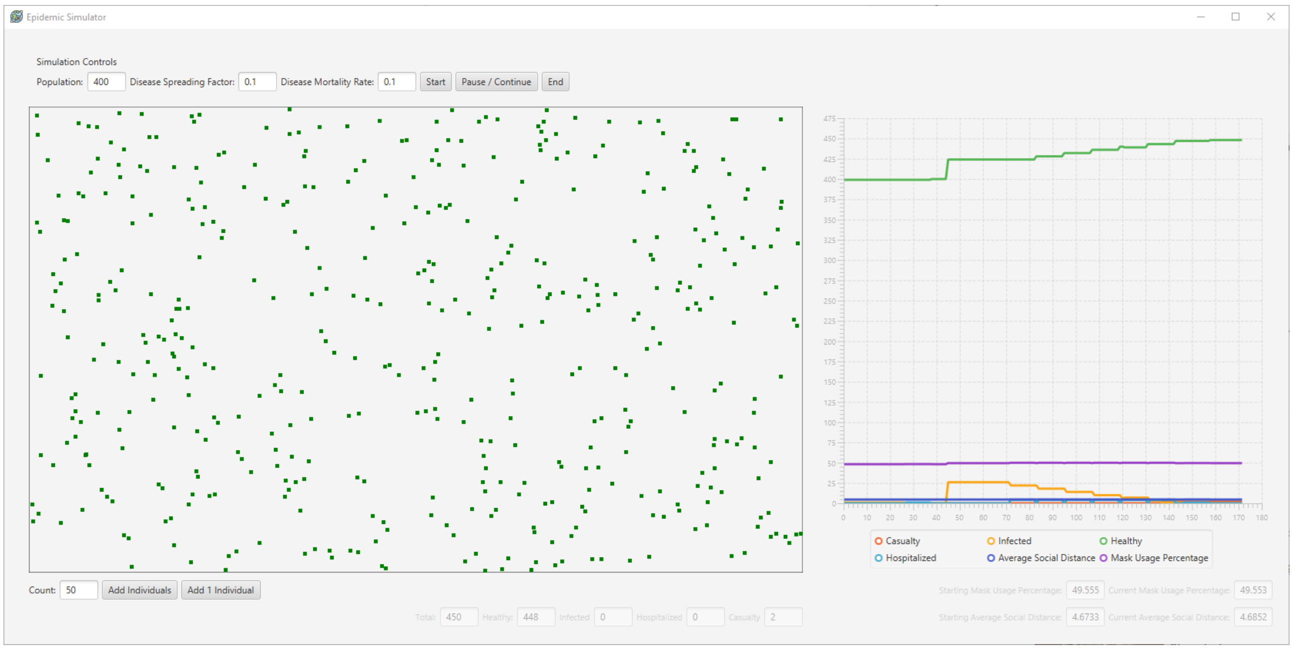
Total: 450 Healthy: 402 Infected: 1 Hospitalized: 0 Casualty: 47

Starting Mask Usage Percentage: 52.666 Current Mask Usage Percentage: 54.342

Starting Average Social Distance: 4.3666 Current Average Social Distance: 4.3746

First observation is, graph change drastically after adding new individuals, its because the first infected person couldn’t manage to spread the infection, because of mask factor and high mortality rate. In this case, hospital is not useful, because casualties occur before they hospitalized due to high mortality rate. To create useful data, random 50 individuals added, almost half of them was infected. It did not triggered an epidemic, because infected individuals died before getting enough chance to spread the disease.

3.5 Scenario 3: 400 / 0.1 / 0.1 with all functionalities


Epidemic Simulator

Simulation Controls

Population: 400 Disease Spreading Factor: 0.1 Disease Mortality Rate: 0.1 Start Pause / Continue End

Count: 50 Add Individuals Add 1 Individual

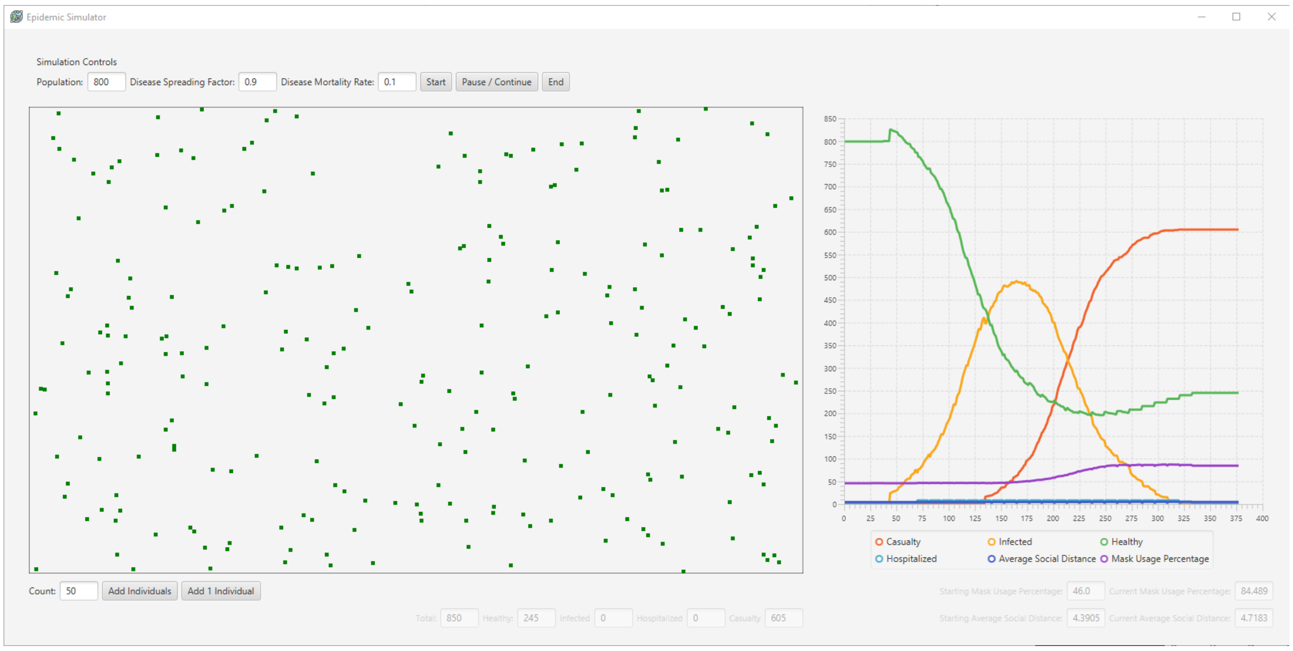
Total: 450 Healthy: 448 Infected: 0 Hospitalized: 0 Casualty: 2

Starting Mask Usage Percentage: 49.555 Current Mask Usage Percentage: 49.553

Starting Average Social Distance: 4.6733 Current Average Social Distance: 4.6852

First observation is, graph change drastically after adding new individuals, its because the first infected person couldn’t manage to spread the infection, because of mask factor and low disease spreading rate. In this case, hospital is very useful, keeps casualty in very acceptable levels. To create useful data, random 50 individuals added, almost half of them was infected. It did not triggered an epidemic, because infected individuals couldn’t spread the disease due to low spreading factor.

3.6 Scenario 4: 800 / 0.9 / 0.1 with all functionalities


Epidemic Simulator

Simulation Controls

Population: 800 Disease Spreading Factor: 0.9 Disease Mortality Rate: 0.1 Start Pause / Continue End

Count: 50 Add Individuals Add 1 Individual

Total: 850 Healthy: 245 Infected 0 Hospitalized 0 Casualty 605

Starting Mask Usage Percentage: 46.0 Current Mask Usage Percentage: 84.489

Starting Average Social Distance: 4.3905 Current Average Social Distance: 4.7183

First observation is, graph change drastically after adding new individuals, its because the first infected person couldn’t manage to spread the infection, probably because of mask factor, with hospital, the infected individual gained its health. To create useful data, random 50 individuals added, almost half of them was infected. It triggered an epidemic. Because of high population, curves were deeper, which means spreading was too fast to deal for hospital, casualty number rise a lot even the mortality rate is low.

Second observation, again, difference in the mask usage percentage between start and end is huge, that can show that casualties are mostly the people without masks, so mask usage has a huge effect on disease spread.

4 Discussion

4.1 Part 5 in Project

4.1.1 Disease Spreading Factor (R)

Disease Spreading Factor plays a crucial role for pandemic, it is one of the main factors of the infected and dead count, effects them exponentially.

4.1.2 Disease Mortality Rate (Z)

Disease Mortality Rate plays important role for pandemic, it is one of the main factors of the dead count, effects it exponentially. But when it is too high, it can effect dead count negatively on big picture, because disease is not spread enough to kill more people on total It is effect on infected count is negatively, when infected individuals died too quickly, disease is not spread enough to continue pandemic status.

4.1.3 Mask Use Percentage

Mask Use Percentage has a crucial role for pandemic, it effects infected count exponentially in negative way. It has effect on casualties too, but not directly.

4.1.4 Average Social Distance

Average Social Distance was expected to have effects on the calculations, but in my tests, it has almost no effect on infected and died counts. Due to random selection of social distance, tests can be misleading.

4.1.5 Population

Population effects infected count exponentially, but it effects casualty count linearly.

4.1.6 Hospital Capacity

Hospital Ventilator Capacity is a very important metric to drop casualty numbers and increase the time for having casualties at start. It also changes the increase/decrease from exponential to linear in healthy count, infected count and casualty count. Behaves like a barrier in front of an avalanche, that will

4.2 Future Development

Refactoring builders or adding new types of builders to create scenarios in more controlled way, is very easy, due to separation of concerns with factory and builder structures.

UI and Simulation is separated completely from each other, like a layered structure, Controller connects them. This way, UI changes doesn’t break the simulation, and simulation can be implemented with another interface because of its API-like structure.

Simulation parameters are mostly controlled from Common class with static fields, this also makes easy to change many things inside the code base.

Mediator class holds every method as static method, this lets us to implement complicated logic in one place and can use these methods all around the program. In some cases, methods has to be inside in some entities, with help of a mediator, these methods will be only wrappers and the main method that holds the logic inside will be in mediator.

Only downside can occur is to making changes in Individual states. Managing infection, health, hospital, alive status of Individuals can be implemented with state pattern by using another FSM, which can help to apply single responsibility principle. From this point it would be better to go that way, but I decided that will increase complexity of code due to having 2 different FSM and increase number of classes, so I did not implement it.

4.3 Feedback About Project

It was a very fun project, that offers a relation with real life and helped me to understand mask usage and having hospital is in such scenarios like we are in right now. Also, it was a good practice for understanding purposes of design patterns and their composite usage. Only downside is, it was a time-consuming project before exams, at least for me it took almost a week with full time work. It would be great if the project 1 week ago, so there can be more time to focus final exams.