VIRUS EVOLUTION

Project Report

SUMMER 2021

INFO6205

Project Group 5

Sunit Bail (001577432)

Jay Devnani (001521319)

Introduction

Aim:

To study the evolution of variants of a positive-sense single-stranded RNA virus.

Approach:

COVID-19 simulation connects two different modules, one that tracks the mutation of a virus within a host or environment, and another that governs its spread to other hosts or environments.

We start with a host population of a 1000, one of whom is infected with a randomly generated Virus strand. By tracking the exact co-ordinates of the infected host and those around them, we identify

1. The next host to be infected
2. The strain(s) that would cause the next host to be infected

This simulation runs for 730 days. The progress of each day is detailed in Island.java.

We begin execution with SImulationDemo.java. We first create the Swing based UI of this project. Once the base is set, we begin a loop that would represent the start of each day in or simulation. This makes use of the Observer-Observable pattern, where the Observable class initiates a new day and the Observer class (in this case Island), executes the actions of the day.

1000 hosts are initiated with one infected individual, with randomly selected co-ordinates to render on the UI as well as to track infections with neighbors.

1. There are four distinct genotypes amongst these hosts that determine their immunity against COVID-19, as a result, determine the fitness value of the virus if they are infected. These genotypes are assigned equally when the hosts are initialized. As a function of the host's genetic makeup, the genotype of an individual is not subject to change.

2. Based on infection status, there are three categories of hosts, those that have never been infected, those that have been previously infected and those who are vaccinated. These factors, in conjunction with the host's genotype, determine the fitness of the virus within a given host. Infection Status is subject to change upon recovery and vaccination.

Program

Data Structures & Classes:

Data Structures used: HashMap, LinkedList

List of Classes:

Constants, GeneticAlgorithm, Host, Island, Virus, VirusSimulation, SimulationResult, SimulationDemo

List of Built-in Classes:

Java.awt.Color, java.awt.Graphics, java.awt.Graphics2d, java.awt.Shape, java.awt.geom.Ellipse2d, java.awt.Dimension, java.awt.event.ActionEvent, java.awt.event.ActionListener java.util.Random, java.util.Hashmap, java.util.Arraylist, java.util.Iterator, java.util.List, java.util.Map, java.util.Observable, java.util.Observer

Javax.swing.BoxLayout, javax.swing.JButton, javax.swing.Jframe, javax.swing.Jpanel, javax.swing.JSplitPane

Algorithm:

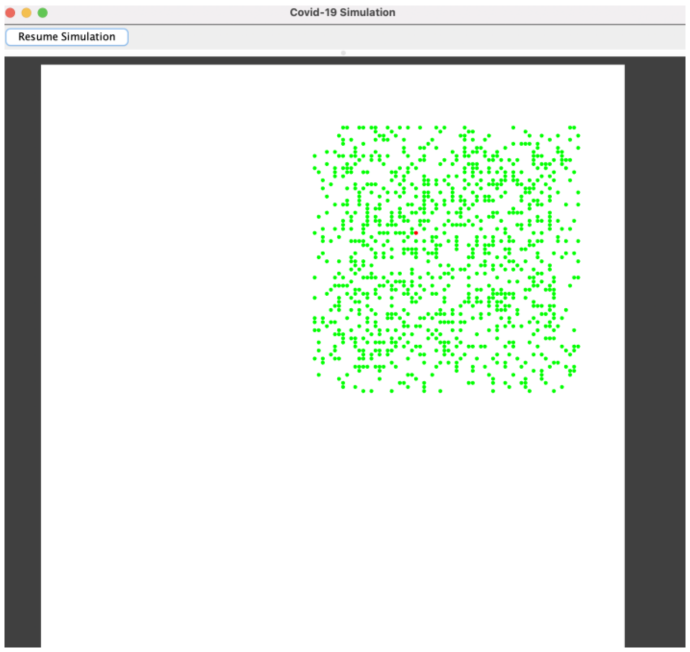
Genetic Algorithm (GA) is a procedure inspired by natural selection. It is an optimization technique used to solve nonlinear or non-differentiable optimization problems. Concepts from evolutionary biology are used in order to search for a global minimum optimization problem. It uses an initial generation of candidate solutions and subsequent generations from this first generation are created using selection, crossover and mutation.

In our project, we have used GA in order to mutate the virus and recombine chromosomes of the host in order to create newer generations of the virus.

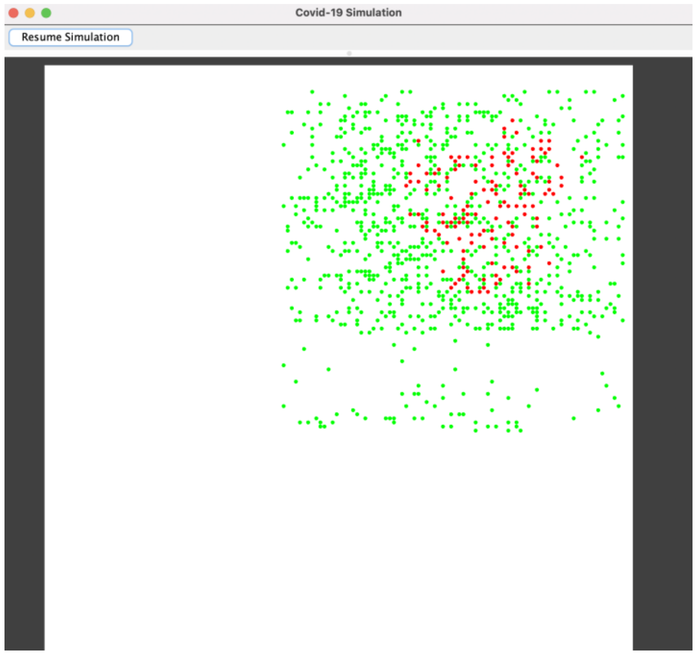
Limitations:

- Mutation rate is kept constant
- Host population is kept constant, newer hosts are not included which is a major factor when it comes to the spread of virus
- Efficacy of the vaccine is not questioned. If a person is vaccinated, they are immune.

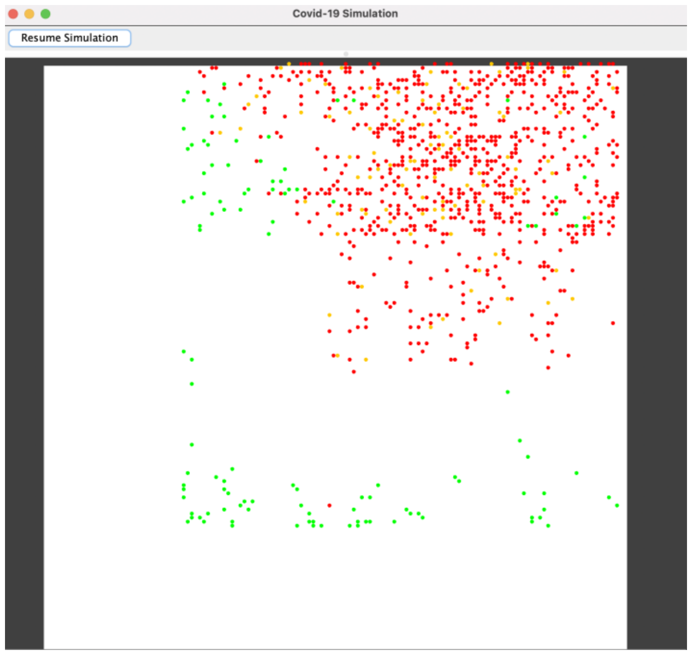
UI Flow



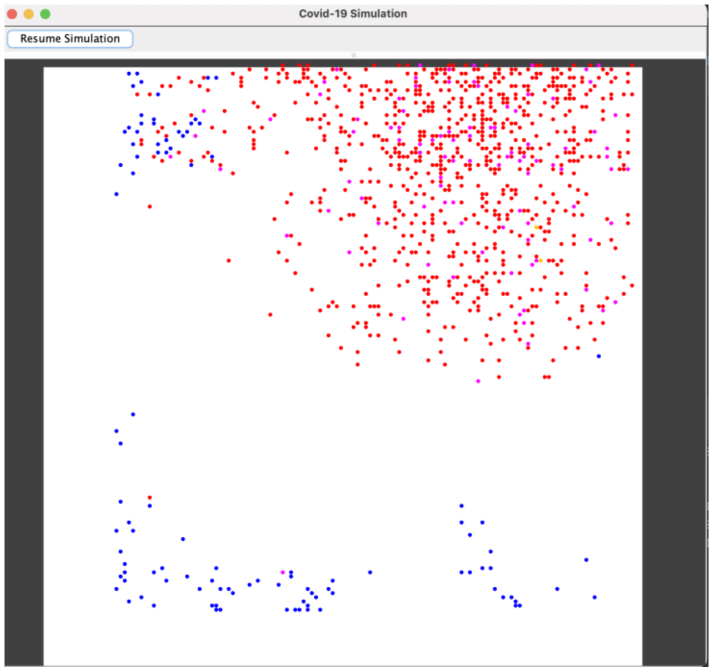
A singular virus (RED) strand is seen in the population of healthy humans (GREEN) during the beginning of the simulation



As can be seen, the virus has now spread and mutated with the chromosomes of other humans

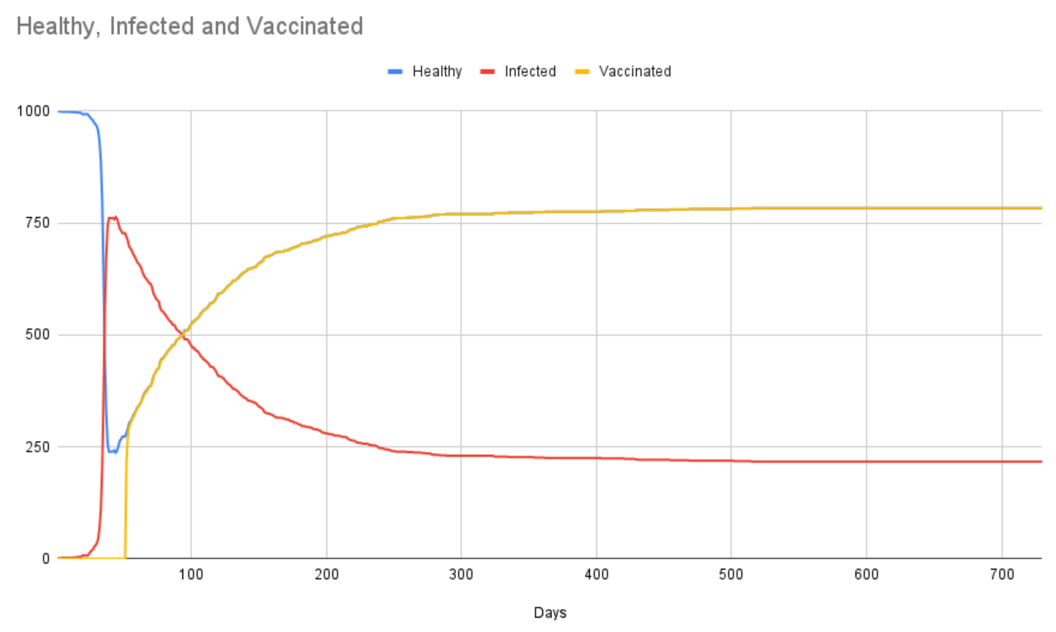


The virus rampantly spreads, leaving only the hosts that maintain their distance to not be infected with some humans even recovering from the virus (ORANGE)



After a certain amount of time, the vaccines have been rolled out- given to those who have not been infected as yet (GREEN -> BLUE) or those hosts who have recovered (ORANGE -> MAGENTA)

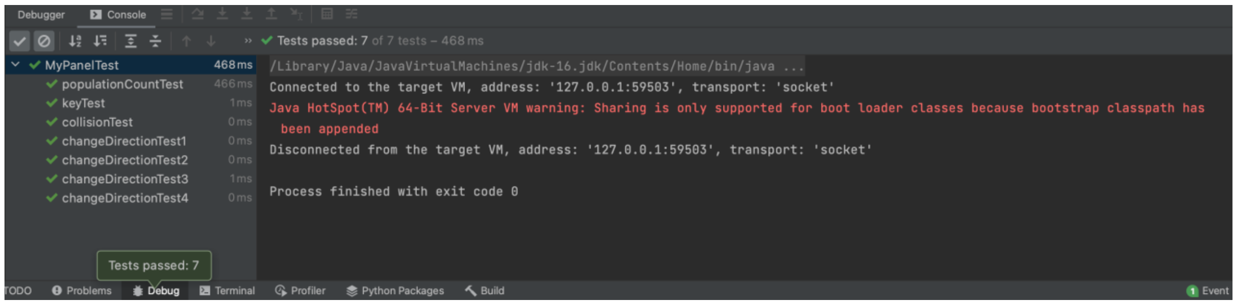
Graphical Analysis



Results:

Please refer to results file in the Github Repository

Test Cases



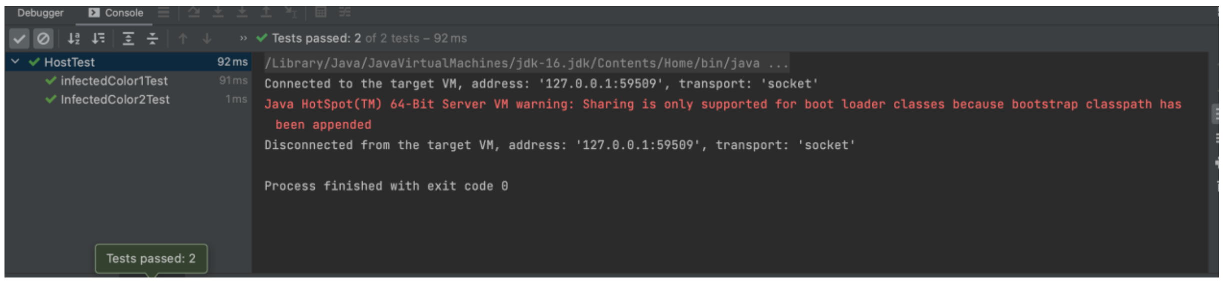
```
✓ Tests passed: 7 of 7 tests – 468 ms

✓ MyPanelTest                 468 ms
    ✓ populationCountTest     466 ms
    ✓ keyTest                   1 ms
    ✓ collisionTest             0 ms
    ✓ changeDirectionTest1      0 ms
    ✓ changeDirectionTest2      0 ms
    ✓ changeDirectionTest3      1 ms
    ✓ changeDirectionTest4      0 ms

Tests passed: 7
```

```
/Library/Java/JavaVirtualMachines/jdk-16.jdk/Contents/Home/bin/java ...
Connected to the target VM, address: '127.0.0.1:59503', transport: 'socket'
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has
 been appended
Disconnected from the target VM, address: '127.0.0.1:59503', transport: 'socket'

Process finished with exit code 0
```



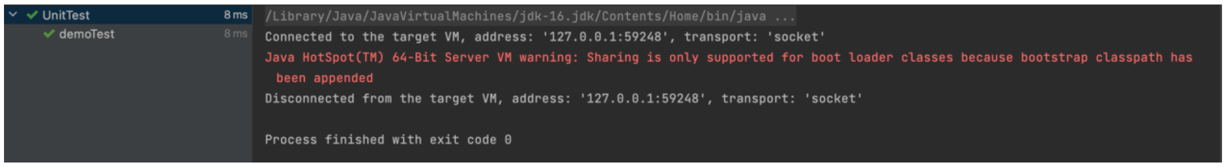
```
✓ Tests passed: 2 of 2 tests – 92 ms

✓ HostTest                    92 ms
    ✓ infectedColor1Test      91 ms
    ✓ InfectedColor2Test       1 ms

Tests passed: 2
```

```
/Library/Java/JavaVirtualMachines/jdk-16.jdk/Contents/Home/bin/java ...
Connected to the target VM, address: '127.0.0.1:59509', transport: 'socket'
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has
 been appended
Disconnected from the target VM, address: '127.0.0.1:59509', transport: 'socket'

Process finished with exit code 0
```



```
/Library/Java/JavaVirtualMachines/jdk-16.jdk/Contents/Home/bin/java ...
Connected to the target VM, address: '127.0.0.1:59248', transport: 'socket'
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has
 been appended
Disconnected from the target VM, address: '127.0.0.1:59248', transport: 'socket'

Process finished with exit code 0
```

As can be seen, all the unit test cases have passed.

Conclusion

From this report, we have successfully performed Viral evolution. Upon running the simulation, different values for each of these metrics were discovered based on the virus's fitness value, the genotype of the hosts and the statistical benchmarks used. The level of randomness discovered upon repeated execution of this simulation with no parameters changed between runs prove that this is an acceptable accurate representation of a randomly mutating virus infecting a host population that has the ability to recover from and be vaccinated against it.

The simulation illustrates the interaction between the diverse types of hosts and their subsequent types of viruses. Once vaccinated, the number of infected hosts slowly stops to climb, but the interaction between the viruses present continues till the simulation is complete. During the simulation, it was also noticed that due to mutations and recombination of the different strains, a new species of the virus had emerged which was heavily present in B1 type hosts who had never previously been infected by the virus and had not been vaccinated yet.

References

- Genetic Algorithm – retrieved from Wikipedia: https://en.wikipedia.org/wiki/Genetic_algorithm

- How to calculate fitness function – retrieved from article dates July 21, 2017 in 'towards data science' : https://towardsdatascience.com/how-to-define-a-fitness-function-in-a-genetic-algorithm-be572b9ea3b4
- Severe acute respiratory syndrome coronavirus 2 – retrieved from Wikipedia: https://en.wikipedia.org/wiki/Severe_acute_respiratory_syndrome_coronavirus_2
- Learning: Genetic Algorithm – retrieved from MIT OpenCourseWare, Instructor: Patrick Winston : https://www.youtube.com/watch?v=kHyNqSnzP8Y
- Understanding mutation and crossover – retrieved from Computerphile, Instructor: Dr. Alex Turner: https://www.youtube.com/watch?v=MacVqujSXWE: