CSE – 443 OBJECT ORIENTED ANALYSIS AND DESIGN

FINAL REPORT

OĞUZHAN SEZGİN
1801042005

My final assignment I use different design patterns. Firstly, when I create individuals, I use **composite** and **factory design patterns**. Because with composite design pattern the program does not care the one element or an array. There is abstract **IndividualComponent** class and it has some methods. These methods throw exceptions. **Individual** and **IndividualArray** classes extends the **IndividualComponet** class. The **Society** class has a **IndividualComponent** class these because user can add more than one individual s/he can add them in bulk as well as one by one. I also use **iterator** design pattern because of the iterate all individuals in society. There is a **CompositeIterator** class. These class iterate all individuals; it does not care the array all single element.

On the other hand, there is a **Factory** interface and **IndividualFactory** class. **IndividualFactory** class implements the Factory interface. These class create individuals and return the user. Because of the individuals must communicate each other with mediator class it has to have a mediator and the factory class create individuals with mediator class.

The individuals and individual arrays store in society class. The society class has society individuals and society information (infected number, healthy number etc.). The individuals has **waitingTime, lifetime, inHospitalTime** and **goHospitalTime** . The **waitingTime** is he waiting time when the two individual collision, lifetime the life time when the individual get infected, **goHospitalTime** is constant time , when the individual get infected s/he goes the hospital these time later, **inHospital** time is the waiting time in hospital. Program checks these times each second and decrease these times.

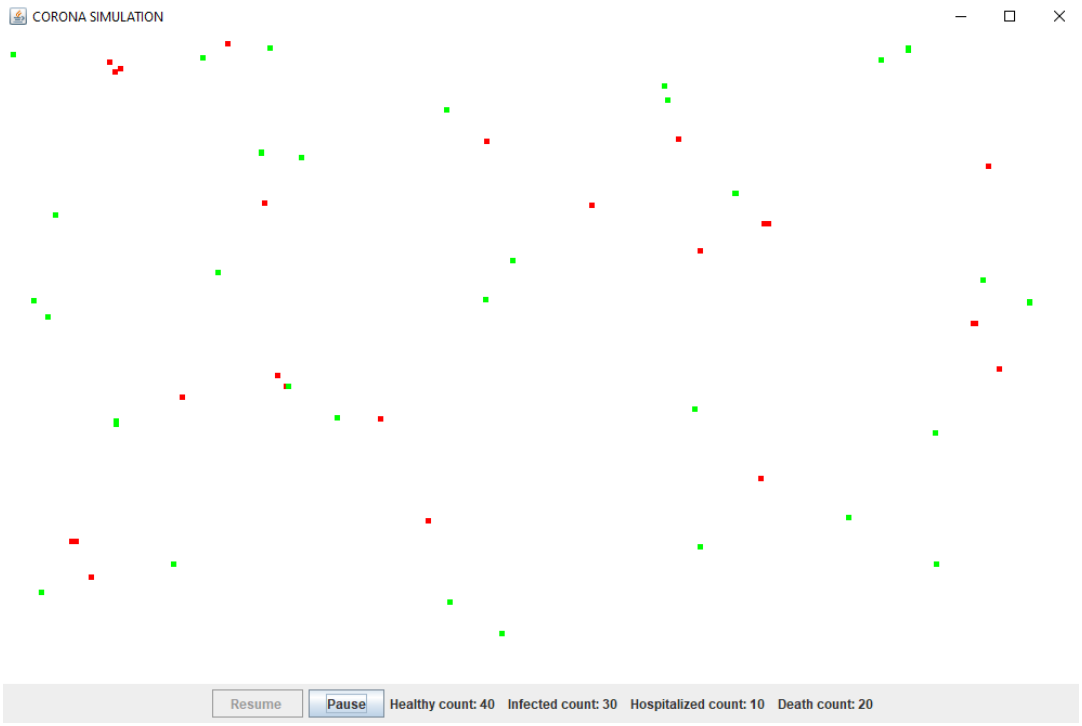
In basis of simulation program, I use most known **compound design pattern MVC.** I have three modules: model, view, controller.

In model I use **observer design pattern**. There is a **SocietyModelInterface** and **SocietylModel** classes. **SocietyModelInterface** is like subject. **SocietyModel** implements **SocietyModelInterface**. **SocietyModel** has Society class. Individuals also store in **SocietyModel** class.These class has observers. When the individuals change their location the **SocietyModel** notifies observer, and observer change view. The observer is **SocietyView** class. **SocietyModel** class has run methods These methods change the location of the individuals and check their status. These thread function sleep 1 second and then change individual's location and status then notify controller according to individual status.

In controller there is a **ContollerInteface** and **IndividualController** class. In this part I use **strategy design pattern**. **IndividualController** class implements the **ControllerInterface**. Implements function according to its own control. **IndividualController** class create a contact with view and the model parts like a bridge. When user press the **pause** or **resume** button, controller transfer request to the model part. It also has hospital and mediator class. When the individuals move, in every move controller call mediator's **chekCollision** method and checks the collision. Also, when the individual must go to the hospital it taken to the hospital with **goHospital** method. If hospital has not enough ventilator it pushes individuals to the **waitingIndividual** and waits the individual in order. If the individual die when waiting order, controller remove the individual from **waitingIndividual** queue. Besides it discharges individuals from the hospital when their time is up.

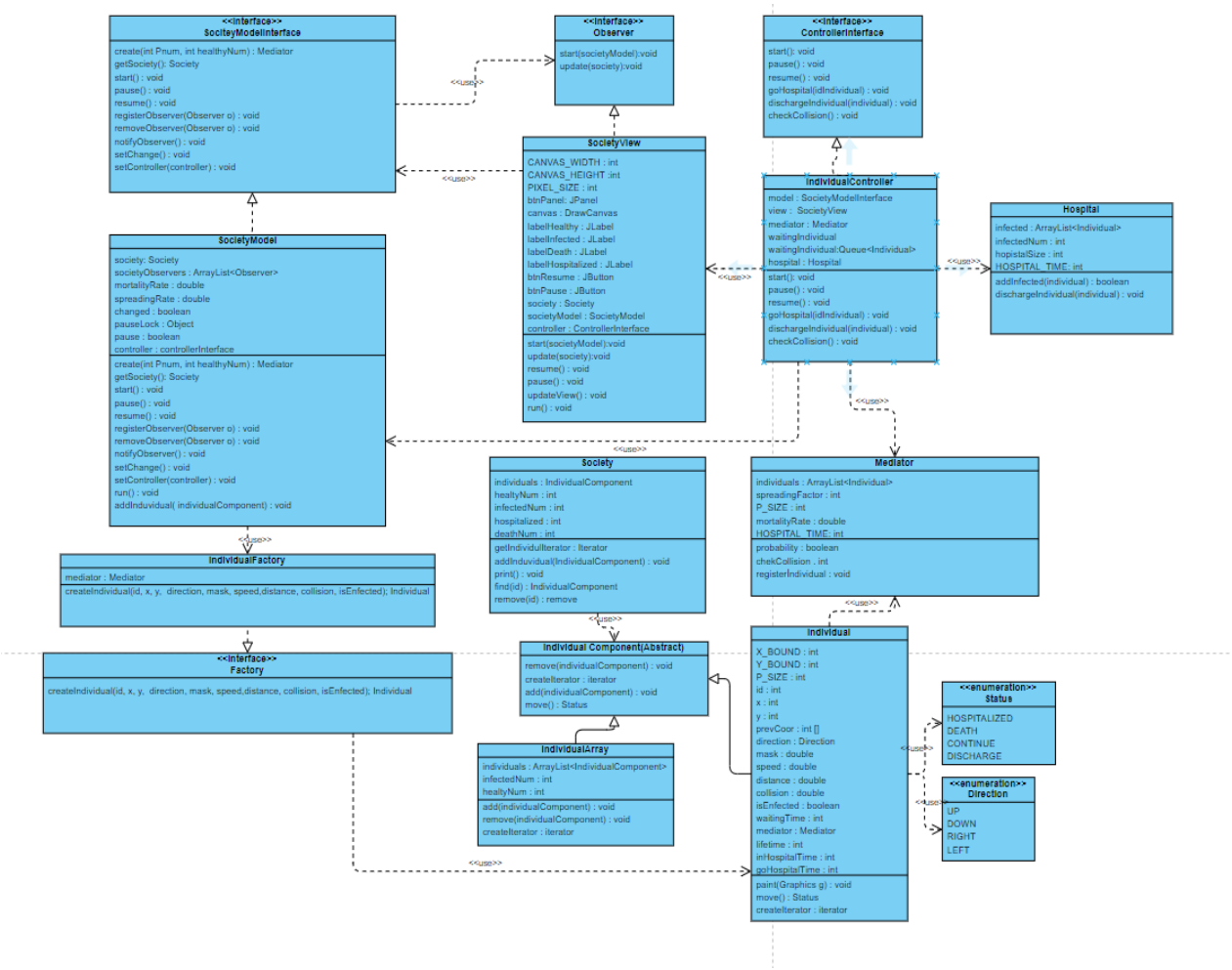
In view model I use **Java Swing**. I add a **canvas**, **pause button, resume button** and **information label.** In this part I also use **composite design pattern** because the view like a

three and it has frame, buttons and labels. The frame does not care the element is button or label or etc. The **SocietyView** extends the observer class. When the model moves individuals notify view and the view class updates canvas. And also, it is multi-threaded, when the user clicks the pause button the simulation pauses and when s/he click the resume button the simulation resumes. The simulation view showing below:



Red points represent infected individuals and the green points represents the healthy individuals.

The class diagram showing below :



Javadoc document is in doc file(index.html).

There is a **SimulationTester** class in the view package. You can run the program with these class.