JAN 2021

CSE 443 OBJECT ORIENTED ANALYSIS & DESIGN

FINAL PROJECT REPORT

AHMET YUŞA TELLİ
151044092
aytelli@gtu.edu.tr

We have 2 packages one for GUI another one for threads and classes.

1. GUI Package

In GUI package, we have 5 windows, these windows are classes.

The SecimEkrani welcomes us. If we want to add individuals ourselves, we must enter “1”. If we want to add individuals automatically, we must enter “2”.

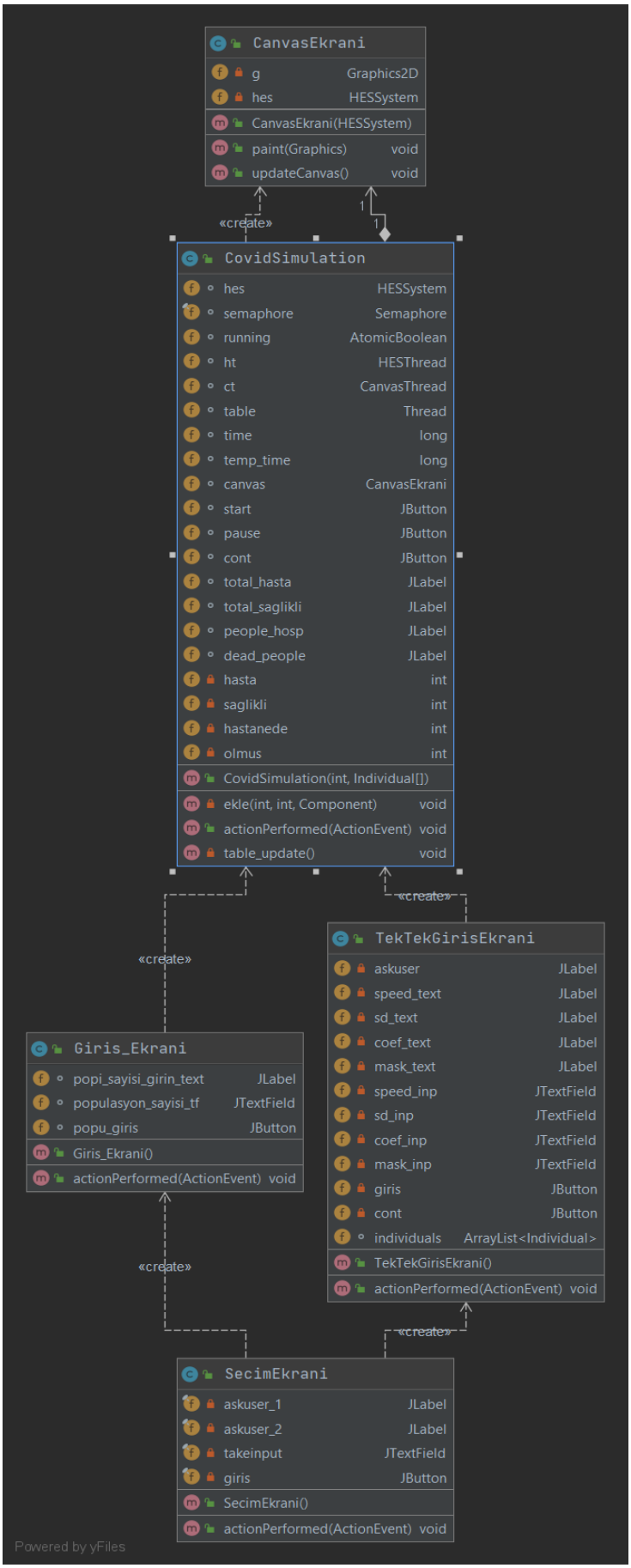
Let’s select “1”, TekTekGirisEkranı opens. We enter 4 features of individual then click enter until our entry process is finished. Then we must click finish button. In this window, the user enters 4 individual features: speed, social distant, coefficient and mask.

If click “Enter” we save them and wait to next individual features. If click “Finish” we send individuals to new window which name is CovidSimulation.

If we select “2”, Giris_Ekrani opens, and asks the number of individuals.

After TekTekGirisEkranı and Giris_Ekrani windows, the CovidSimulation window opens. We have one canvas, some text fields, buttons, threads. We design the screen jbottons, jtextfields etc. Then we create HESSystem object, a thread, a CanvasThread, a HESThread objects. If we select 2 at the beginning, we give the number of populations to HESSystem object. If we select 1 and add some individuals, then we give individuals to this object.

The CovidSimulation is our main window. We have a normal Thread object and two classes object which are extends from Thread. Totally we have three threads. The first thread is updating the text fields; number of individuals, number of dead, number of people in hospital.



Other thread is runs on HESSystem class which name is HESThread. This thread moves or individuals. Then, check their collides and send to the hospital.

The last thread is runs on Canvas class which name is CanvasThread. This thread repaints our canvas.

We draw individuals on the canvas. One is patient others are healthy.

There are three buttons: start, pause and continue. When we click start, three threads start: CanvasThread updates canvas, HESThread move individuals and table thread updates the number of individuals, healthy people, dead and patients.

All threads wait one second after running. So, the program runs in real time. Patients move on canvas before 25 seconds then go to the hospital.

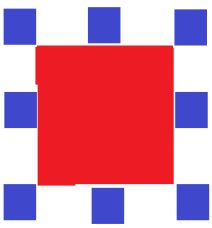
2. Final Project Package

We have three classes which are extends from thread. They are CanvasThread, HESThread and HastaneThread. CanvasThread and HESThread are used in CovidSimulation. HastaneThread is used in HESSystem. It checks patients in hospital and if they are more than 10 seconds in hospital remove them and take other patients which are not in hospital.

HESSystem class is our Mediator class. We control all individuals in this class, and we simulate the canvas using two-dimension array its name is zombieland. First, we give zeros to the zombieland, after individuals create, we write their IDs to this zombieland in their coordinates. We control all individuals using this array. After moving we check collisions. We have a Hospital object, a HospitalThread object.

We move individuals in this class. We take one individual and select random way left, right, up or down and move it. Then, we update our zombieland. After all individuals moves, we check collision. If one patient is nearby other person, we calculate probability.

How we check collision? A person has 5x5 pixels. We check eight pixels nearby it. See in this picture. The red is a patient. And we check the blue pixels, if these pixels have some ID from other people, we check the probability of the healthy people being sick. If probability is greater than 0.25, the healthy person gets sick. After that, we check patients and their sick times. If a patient fell ill 25 seconds ago, we send it to the hospital. HESThread is doing all this.



Red is patient, check the blue pixels.

In Hospital class, we take two different arrays. One for people in hospital which name is hastanedeki_hastalar. Other one for sick people but not in hospital we call them zombie and the array is zombie_hastalar. If hospital capacity is enough, we add to hastanedeki_hastalar array but it is full we add zombie_hastalar. HastaneThread is doing all this. Take to the hospital means, we add to our array and start its time and remove from canvas.

Zombie_hastalar array means, if hospital is not enough for all patients, we add this array but do not remove from canvas. But we start its time.

In zombies we check their time for dead. If their time is greater than 100 * (1-Z), it will die.

In individual class, we generate random individual. The person who ID 1, is sick person others are healthy.

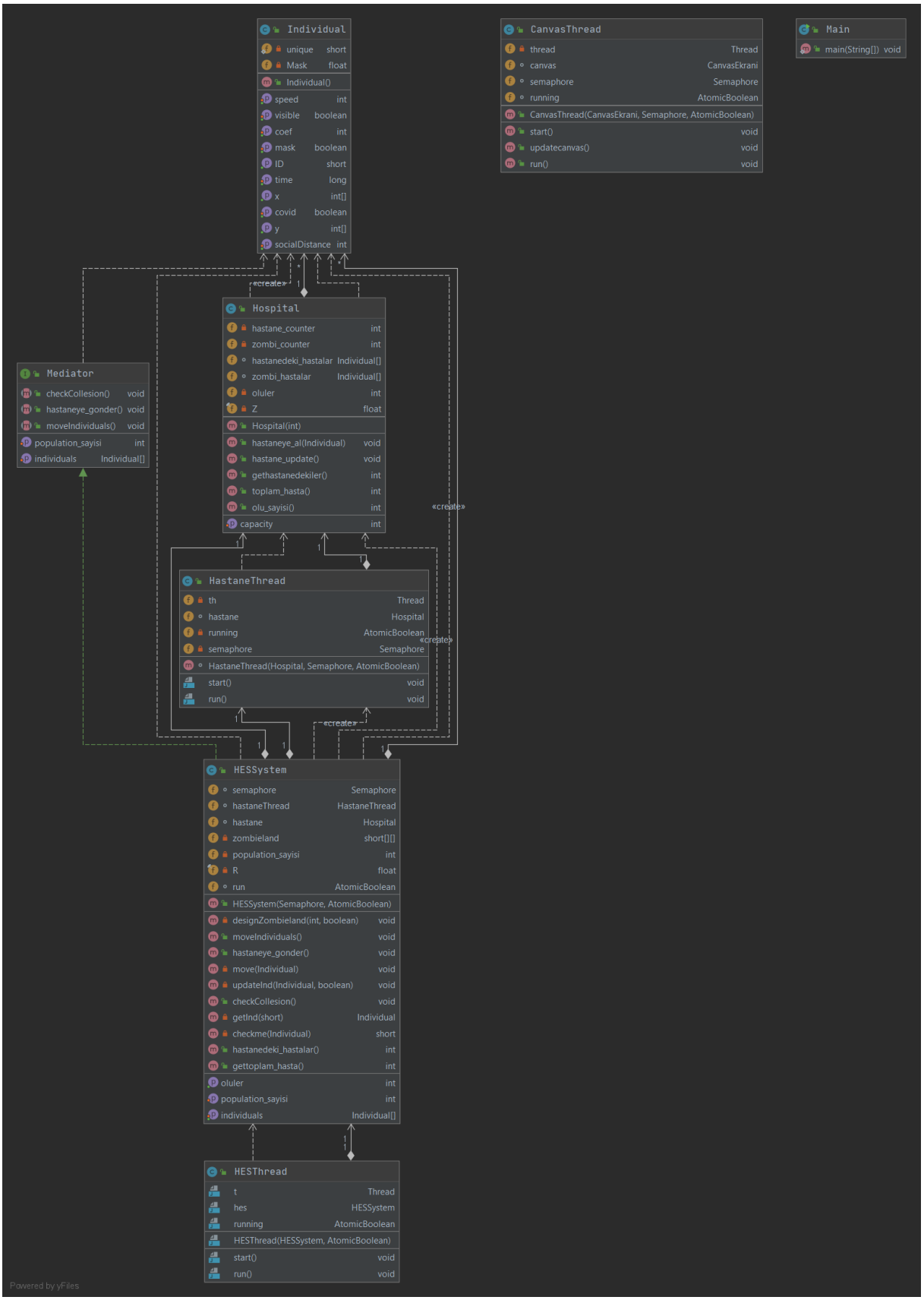
This HESSystem is producer and, the hospital is consumer. In the system, create some patients and hospital take them. But the hospital does not check the patients before 25 seconds. So, we solve the producer consumer problem.

We tried to use observer pattern for this hospital. But it is not completely true. We implemented mediator pattern.

Class diagram is in last page.

NOTE 1: We do not check C time for collision. They do not wait C seconds.
NOTE 2: We do not plot any graphs.

Ahmet Yuşa Telli
151044092


Individual
unique short
Mask float
Individual()
speed int
visible boolean
coef int
mask boolean
ID short
time long
x int[]
covid boolean
y int[]
socialDistance int
CanvasThread
thread Thread
canvas CanvasEkrani
semaphore Semaphore
running AtomicBoolean
CanvasThread(CanvasEkrani, Semaphore, AtomicBoolean)
start() void
updatecanvas() void
run() void
Main
main(String[]) void
«create»
Hospital
hastane_counter int
zombi_counter int
hastanedeki_hastalar Individual[]
zombi_hastalar Individual[]
oluler int
Z float
Hospital(int)
hastaneye_al(Individual) void
hastane_update() void
gethastanedekiler() int
toplam_hasta() int
olu_sayisi() int
capacity int
Mediator
checkCollesion() void
hastaneye_gonder() void
moveIndividuals() void
population_sayisi int
individuals Individual[]
HastaneThread
th Thread
hastane Hospital
running AtomicBoolean
semaphore Semaphore
HastaneThread(Hospital, Semaphore, AtomicBoolean)
start() void
run() void
HESSystem
semaphore Semaphore
hastaneThread HastaneThread
hastane Hospital
zombieland short[][]
population_sayisi int
R float
run AtomicBoolean
HESSystem(Semaphore, AtomicBoolean)
designZombieland(int, boolean) void
moveIndividuals() void
hastaneye_gonder() void
move(Individual) void
updateInd(Individual, boolean) void
checkCollesion() void
getInd(short) Individual
checkme(Individual) short
hastanedeki_hastalar() int
gettoplam_hasta() int
oluler int
population_sayisi int
individuals Individual[]
HESThread
t Thread
hes HESSystem
running AtomicBoolean
HESThread(HESSystem, AtomicBoolean)
start() void
run() void
Powered by yFiles