1)

Object Oriented Programming

Ex0

Part A

: By
Hagai Hen 313414872

Nir Geron 315874925

2)

1) Literature:

1. https://github.com/00111000/Elevator-Scheduling-Simulator
2. https://github.com/joeblau/sample-elevator-controlsystem/tree/master/src/main/java/com/joeblau/ecs
3. https://www.geeksforgeeks.org/smart-elevator-pro-geek-cup/
4. https://www.youtube.com/watch?v=siqiJAJWUVg&ab_channel=ThinkSoftware

2) **The difference between offline and online**

As we start with an offline scenario, we have the whole requests in the beginning so we can plan our program and optimize it by seeing the whole picture.

In the case of an online scenario, we need to write an algorithm that gets a live request and immediately chooses an elevator.

Offline algorithm

In the beginning, our algorithm solves the problem by separating the missions into two different tendencies and floors area, by dividing the problems into areas and tendencies we can reduce the burden of the mission and make optimization which makes it quicker.

When we get all the requests we divide the areas and tendencies according to the number of elevators.

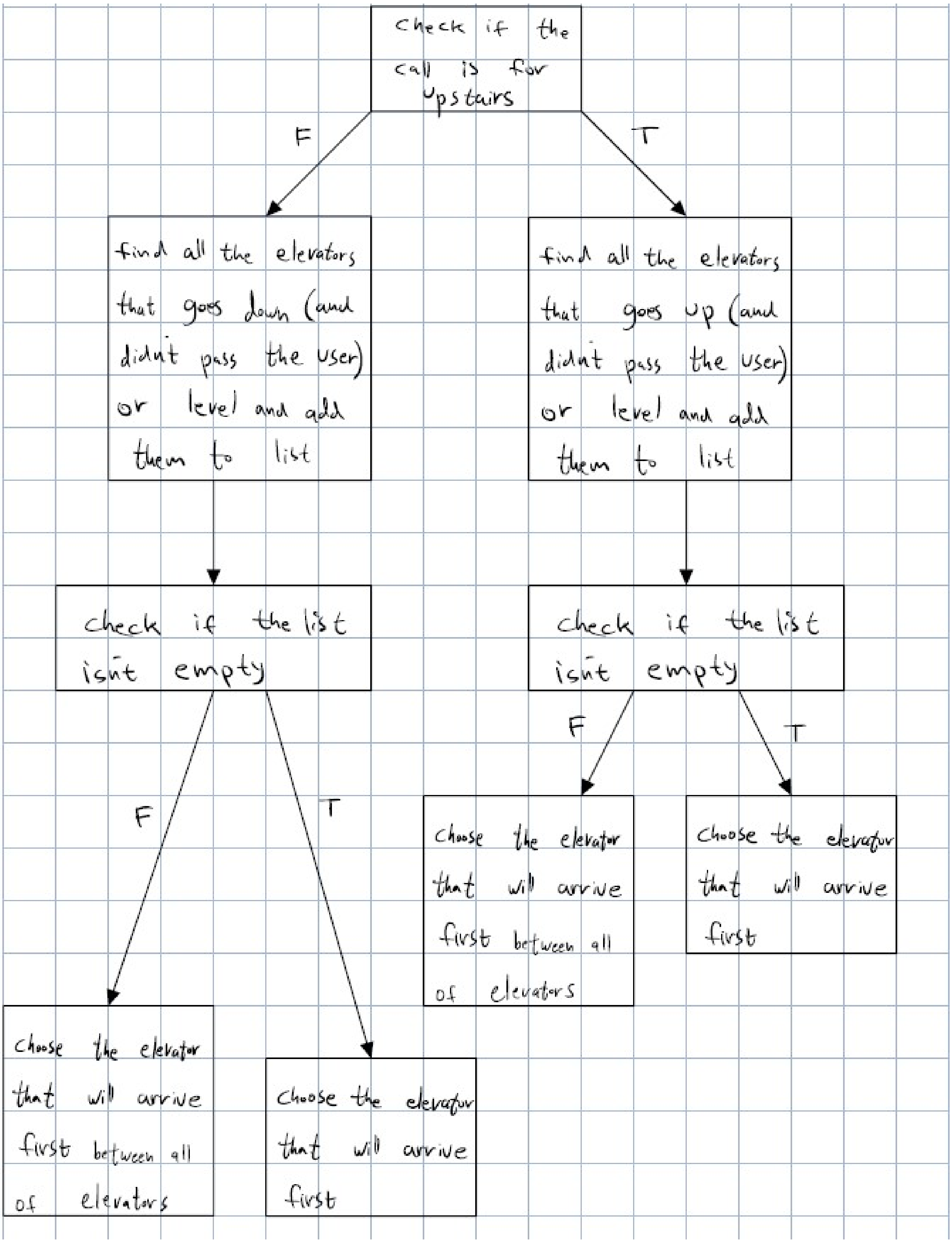
Every time that elevator reaches a floor she's checking if she got another request in her specific location with the same floors area target or less within the next 5 seconds.

For example, if the case is in the morning hours and 70% of the requests are for upstairs so 70% of the elevators will be assigned for going up and that elevator who chose will divide into a different area.

After finishing the current mission, each elevator switches its tendency according to the requests.

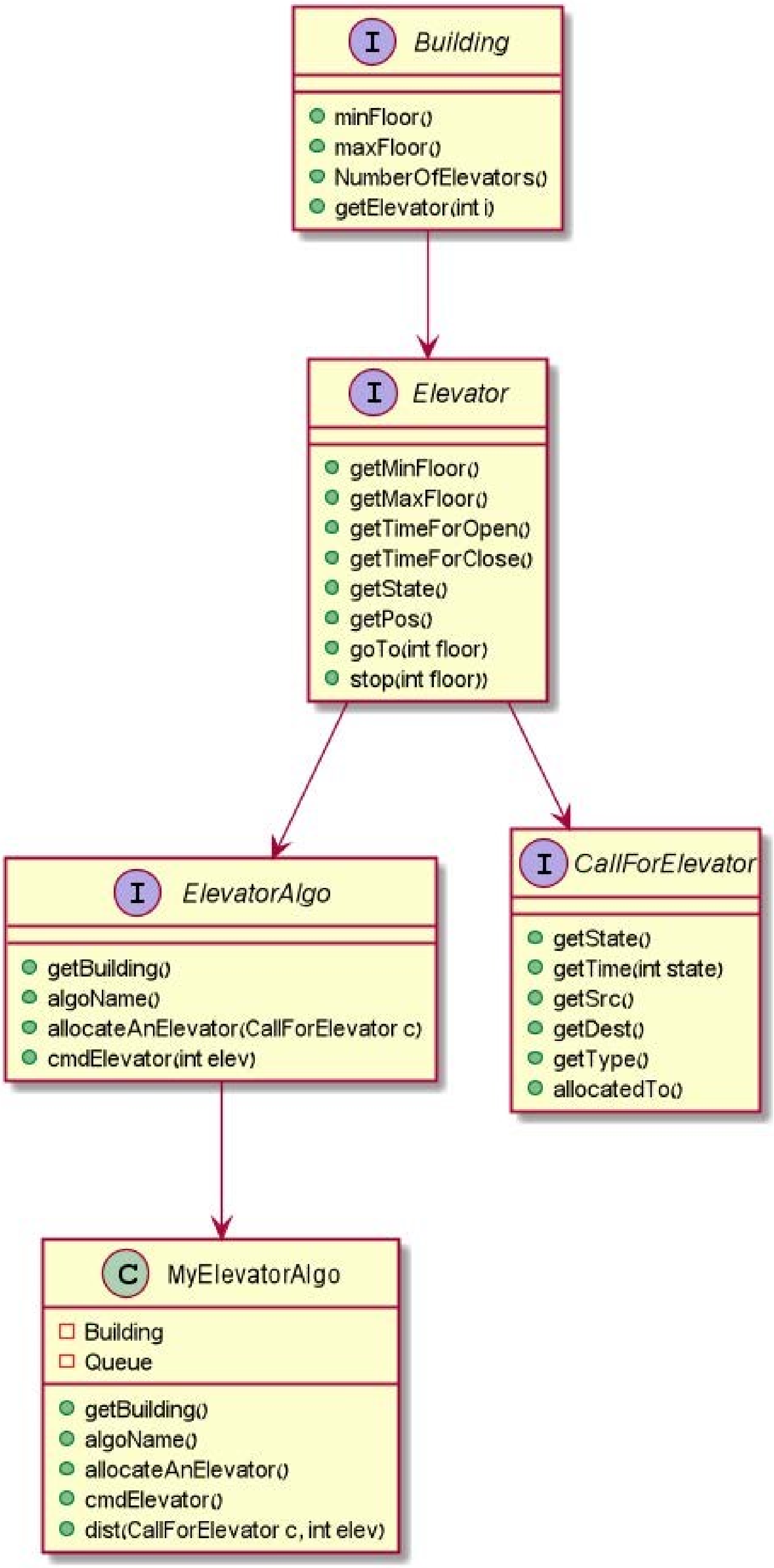
3)

Online Algorithm


check if the call is for upstairs
F
T
find all the elevators that goes down (and didn't pass the user) or level and add them to list
find all the elevators that goes up (and didn't pass the user) or level and add them to list
check if the list isn't empty
check if the list isn't empty
F
T
F
T
Choose the elevator that will arrive first between all of elevators
Choose the elevator that will arrive first
Choose the elevator that will arrive first between all of elevators
Choose the elevator that will arrive first

4)

Classes diagram



5)
In our algorithm we'll check a few functions:

allocateAnElevator:

- We give the function an elevator call and we suppose to get an integer between 0 to the number of elevators.
- Enter a floor that doesn't exists, and check if returns number that exists.

cmdElevator:

- We make an elevator call, for example, a call from 9 to 1, then we'll check if the elevator reaches to his destination.
- Enter a floor that doesn't exists, and check if the elevator goes there (shouldn't)

CalculateTime:

Enter a few calls and check if the time that calculate is what suppose to do, according to the algorithm.

removeAllAppears:

Initial a new ArrayList with duplicates values, after the use of the function, check that this value doesn't exists.

6)

Case	Total waiting time	Average waiting time	Number of incomplete calls	Certificate
0	217.989	21.798	1	-853409955
1	357.989	35.798	4	-1579032365
2	8319.792	83.197	7	-3012301696
3	30789.538	76.973	4	-2246861240
4	35702.455	71.404	5	-2285098599
5	247900.121	247.900	104	-45297312013
6	132000.882	132.000	30	-4612540354
7	492655.121	492.655	247	-4803172600200
8	374565.882	374.565	152	-370936257073
9	97764.340	97.764	18	-3505213947