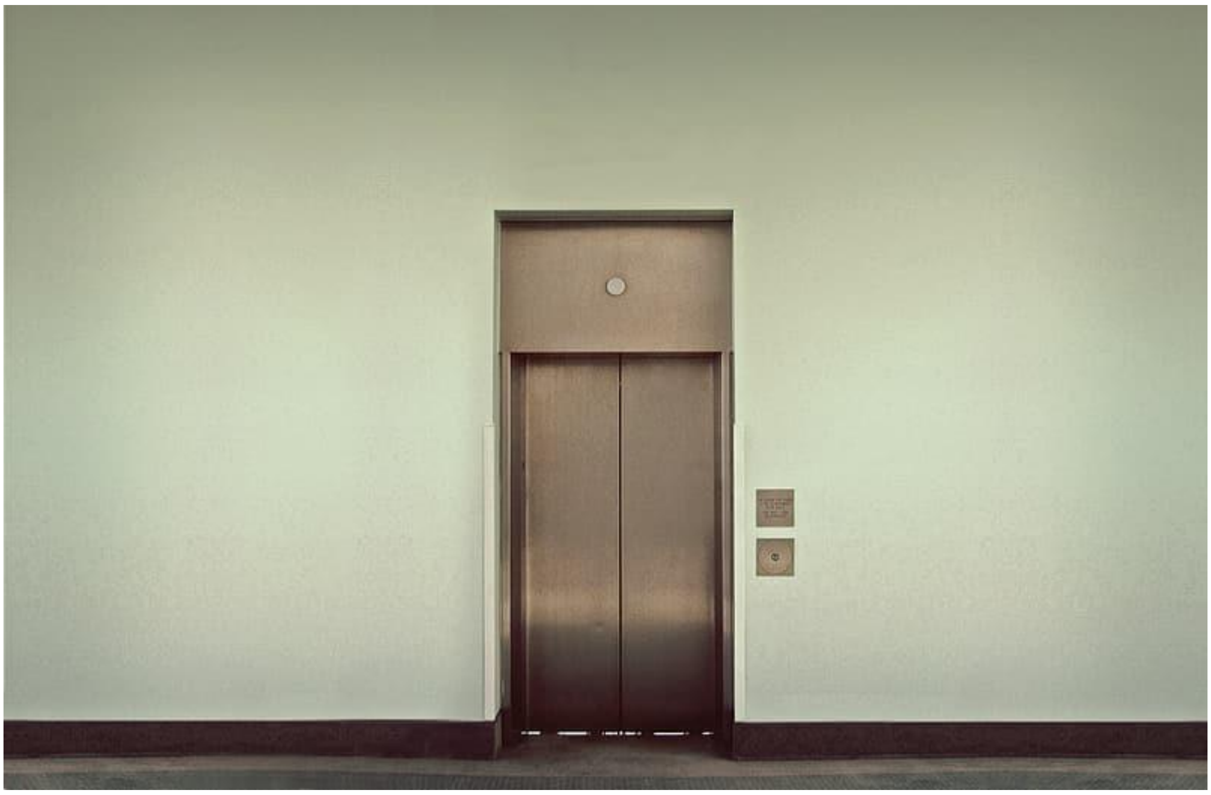
Assignment 0 (Object oriented programming)

Part A



Submitted by:

Bar Goldenberg (209894286)

Sappir Bohbot (316416429)

Literature research:

1. https://www.geeksforgeeks.org/smart-elevator-pro-geek-cup/
2. https://www.youtube.com/watch?v=xOayymoIl8U&ab_channel=SpanningTree
3. https://www.youtube.com/watch?v=siqiJAJWUVg&ab_channel=ThinkSoftware

The difference between an offline algorithm and an online algorithm:

The main difference applies to the knowledge of future input and how the algorithm makes choices in consequence to those inputs which effect run time efficiency, in an offline algorithm we know the inputs ahead of time so we can strategize a game plan to minimize the time it takes to finish the task, in an online algorithm, we receive the input in real-time, therefor the algorithm can only make decisions with the knowledge it has in this point in time.

In consequence, offline algorithms are usually more efficient than online algorithms.

Offline elevator algorithm:

1.Calculate the time it takes to execute the first call that hasn’t been executed, we will define it as T1.

T1=(Destination) * [(Distance to floor) / (Velocity of elevator)] + (Number of opening doors) * (Opening time) + (Number of closing doors) * (Closing time)

2.Find the next call time in the same direction and calculate the difference between the first call and the second one, we will define that number as T2.

```
3.While(true){
For(go over all elevators){
T1=step 1
T2=step 2
    While(T1<T2){
        Add call to elevator queue
        T1=T1+(Time it takes to execute the next call in the same direction)
        T2=Next call time-first call time
}
   Queue.Sort; #sort queue holding elevator calls.
  Elevator.run; #send elevator to calls using the queue.
 }
}
```

Online Algorithm:

First, we build a special Data structure called "smartQueue". It includes two Queues and a pointer.

First queue: “UpQueue” this queue holds elevator calls that are sorted from the lowest floor that was called to the highest.

It is used for calls that are going in the UP direction.

Second queue: ”DownQueue” this queue holds elevator calls that are sorted from the highest floor that was called to the highest.

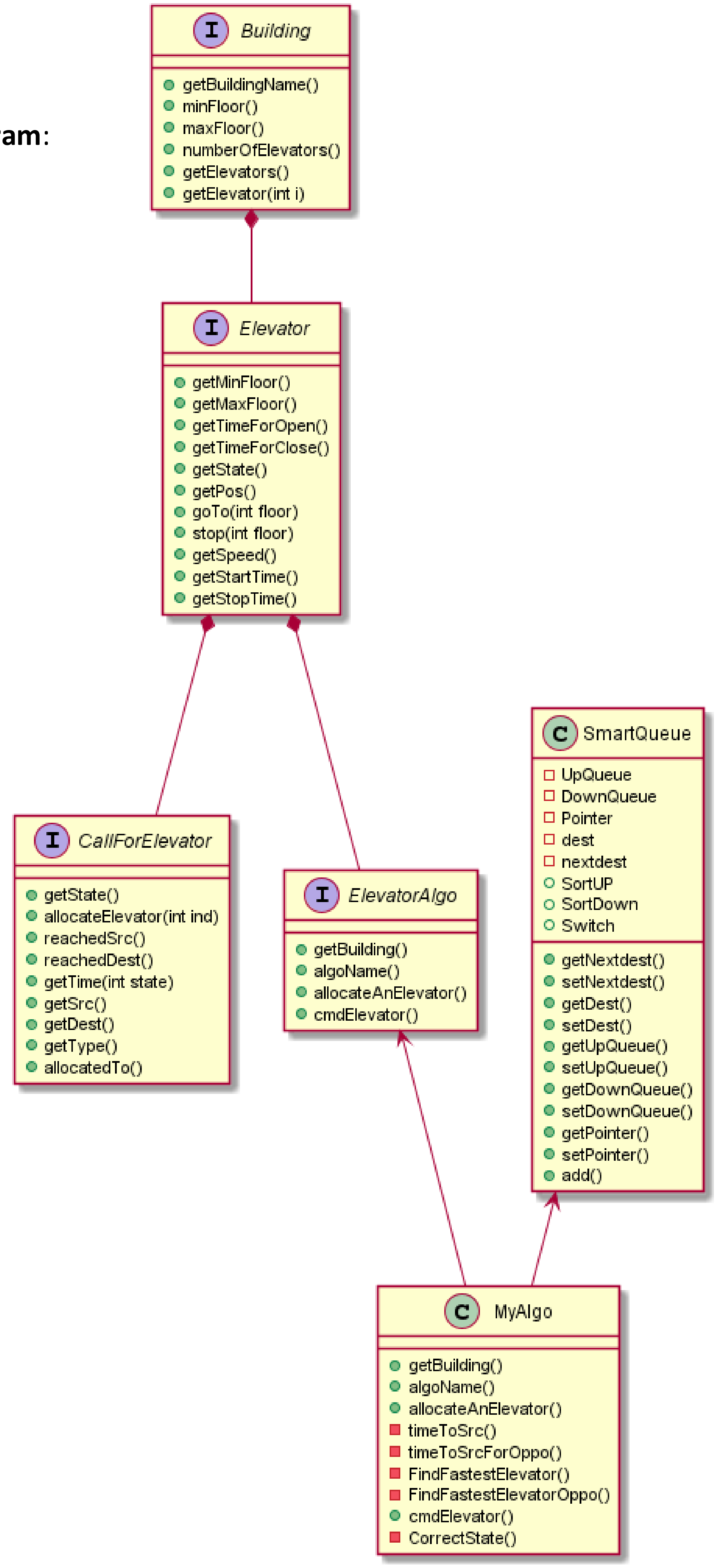
It is used for calls that are going in the DOWN direction.

This data structure will keep all of the elevator stops, and sort them according to the direction (so the elevator won't miss any calls but still operate in the most efficient way).

Now the algorithm called "elevAlgo" includes an array of smartQueues and its size is the number of elevators in the building (so there is one smartQueue for each elevator – It will save all the calls allocated to **this** elevator and arrange them in the most efficient order).

When a new call has been made, the algorithm will first try to allocate the fastest elevator to the source whose position is *Level* or going in the same direction as the call (Elevator must be positioned in a way that we can pick up the call without changing direction) if all elevators don’t meet the requirements, it will allocate the elevator that going in opposite direction and factor there farthest destination to find the fastest elevator.

Class Diagram:



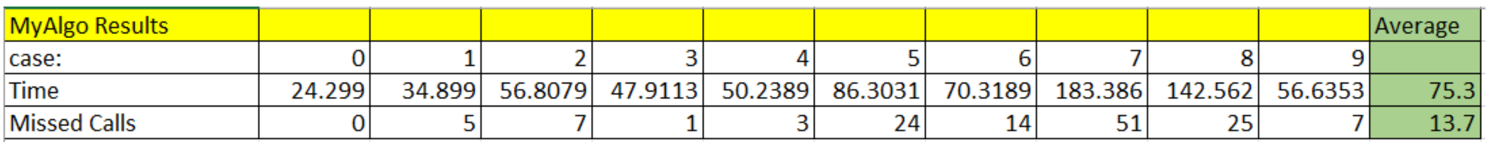
Tests:

We can set all buildings at the same floor, and check if the algorithm picks the fastest elevator.

We can check that no one waits more than two minutes.

Check that a call is received by the elevator.

Results:



MyAlgo Results											Average
case:	0	1	2	3	4	5	6	7	8	9	
Time	24.299	34.899	56.8079	47.9113	50.2389	86.3031	70.3189	183.386	142.562	56.6353	75.3
Missed Calls	0	5	7	1	3	24	14	51	25	7	13.7