Building Design

Updates are highlighted in yellow

This is a complicated project. The purpose of this design step is to help you succeed in this project. We have asked you to build a UML diagram of the entire class structure.

Include the UML Question 9 and 10 will assess this.

Answer the following questions in this document and upload with your UML diagram.

1) How are you storing your elevators in your Building model.

Elevators are stored within the Building model in a List<Elevator> structure. This allows for management of elevators, accommodating buildings with list of elevators. Each Elevator instance within the list represents an individual elevator, containing its own logic for handling requests.

2) How are you storing the incoming requests so you can distribute them to the elevators.

Incoming requests are stored in two separate List<Request> structures in Building: one for up requests and one for down requests.

3) How are you distributing your downRequests and your upRequests to the elevators?

building.addRequest(requestOne)

building.getElevators().get(index).processRequests(requests);

First, use the addRequest method to receive requests in the Building object.

Then, get the list of elevators and then get one of the elevator by its index, so we get the elevator object.

Finally, I use the method processRequests in elevator class to distribute downRequests and upRequests to the elevators.

4) How are you removing all requests when a takeOutOfService request is received.

When Building class implementing stopElevatorSystem method, takeOutOfService is executed on every elevator in the List<elevator> in the building class.

5) How does your step method handle updating the elevators?

If the elevator is out of service then we implement stepOutOfService() method

If the door is open we call the stepDoorOpen function

If the elevator is at the top or bottom we call the stepTopOrBottom function

If we are at the bottom and the direction is down, we need to set the direction to up, we set the timer to this.stopWaitTimeTotal

If we are at the top and the direction is up, we need to set the direction to down, we set the timer to this.stopWaitTimeTotal

If we are not out of service, the door is closed

there is no request at this floor,

we are not at the top or the bottom,

we need to move the elevator in the direction it is currently moving.

6) How do you start processing requests?

First, I put all the requests in a List<Request> as input for processRequests method in class Elevator.

Then, same as (3), use building.getElevators().get(index).processRequests(requests);

building.getElevators().get(index).step();

Finally, implement step function on each elevator instance received requests

7) How do you take the building out of service?

Use stopElevatorSystem method in class Building

If the system is not running, there's no need to stop it

change the system status to stopping

Clear all the recorded requests in building

Clear all the recorded requests in elevators,

Elevators stop taking requests,

directions change to down,

all elevators are out of service,

the timers are set to 0

all elevators keep going down until the elevators reach the main floor it opens the door

all elevators are changed to out of service

8) How do you take the elevators out of service?

By using stopElevatorSystem method in class Building, create a for loop to let every individual elevator out of service by using takeOutOfService() method in class elevator.

In takeOutOfService() method, it clear all the request in this individual elevator, change takingRequests to false, change direction to DOWN, change outOfService to true, change stopWaitTimeLeft to 0. So it guarantees that elevator cannot receive request and change the direction immediately to down, heading to the ground floor and open the door upon arrival altering implementing step() function on every individual elevator.