Building Design

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 in the *List<Elevator>* within the Building class. This list is initialized in the Building's constructor and is designed to hold multiple elevator objects up to the number specified during the construction of the building.

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

Incoming requests are stored in a *List<Request>* in the Building class. The Building class provides methods to add requests to this list, which can then be distributed to the elevators based on their current state and position.

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

Distribution of *downRequests* and *upRequests* can be handled by a method within the Building class, in the *distributeRequests()*. This method would sort requests

into up and down based on the requested start and end floors, then assign them to elevators according to the elevator's current direction, position, and capacity.

Although, at this moment, the algorithm of the rules of how to distribute requests is not figured out yet.

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

The Building class will trigger the *takeOutOfService()* method for the targeted Elevator(by elevator ID) object. This action will prompt the Elevator to change its status, signaling that it is no longer operational and will not accept further requests. At meantime, the queue of requests will be cleared. By these action, all the requests are removing.

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

The step() method in the Elevator class advances the elevator's operation by one time unit. This method updates the *currentFloor* based on the direction the elevator is moving. It checks the *floorRequests* to determine if the elevator should stop at the next floor and manage the *doorStatus*, opening the doors if a stop is needed. If the door is open, the *doorOpenTimer* is decremented until it closes.

6) How do you start processing requests?

To start processing requests, the Building class would utilize a method such as *startElevatorSystem()*. This method sets the status of the *ElevatorSystemStatus* to *RUNNING*, indicating the system is active. It then begins to distribute the requests to appropriate elevators by invoking the *distributeRequests()* method. Each elevator,

upon receiving its assigned requests, will then process them according to its *processRequests()* method, which is detailed in the ElevatorInterface.

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

Using the method to send requests that sets the building's status to OUT_OF_SERVICE and then iterates over each Elevator object within the elevators list, invoking their *takeOutOfService()* method. The elevators is set to status of *STOPPING* proceed to the ground floor, open their doors, and cease to accept new requests. All the elevators won't start until start request is send.

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

Taking an elevator out of service is conducted via the *takeOutOfService()* method defined in the *ElevatorInterface* and implemented by the Elevator class. When this method is invoked by the Building class, the specific Elevator instance will change to its *STOPPING* status, clear any pending requests, and initiate the procedure to move to the ground floor if it is not already there. Upon reaching the ground floor, the elevator's direction is set to *STOPPED*, and the *doorStatus* becomes *OPEN*. The elevator will remain in this state, not participating in the *distributeRequests()* operation, until the *start()* method is called to resume service.