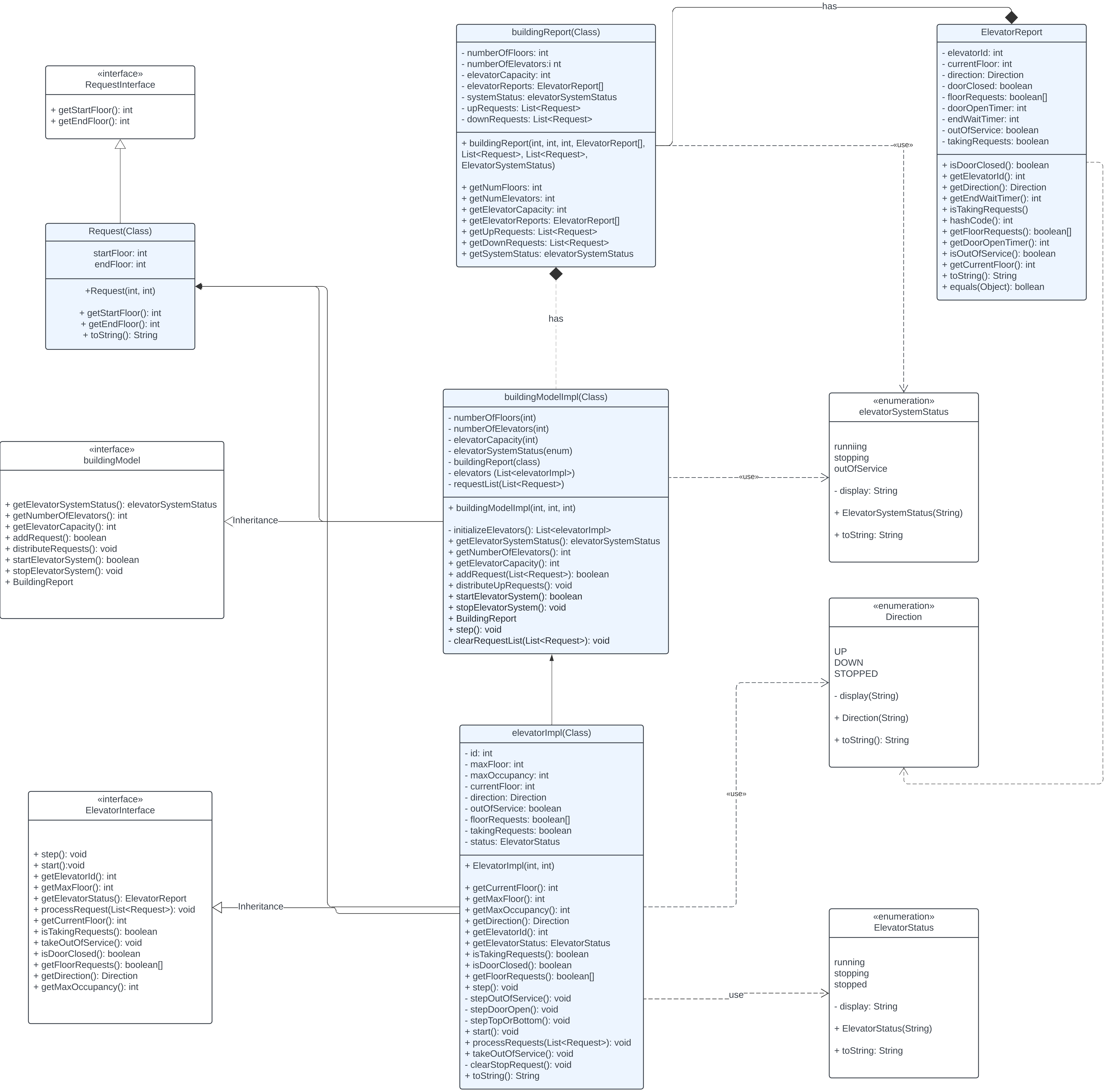

buildingReport(Class)

- numberOfFloors: int
- numberOfElevators:i nt
- elevatorCapacity: int
- elevatorReports: ElevatorReport[]
- systemStatus: elevatorSystemStatus
- upRequests: List<Request>
- downRequests: List<Request>

+ buildingReport(int, int, int, ElevatorReport[], List<Request>, List<Request>, ElevatorSystemStatus)

+ getNumFloors: int
+ getNumElevators: int
+ getElevatorCapacity: int
+ getElevatorReports: ElevatorReport[]
+ getUpRequests: List<Request>
+ getDownRequests: List<Request>
+ getSystemStatus: elevatorSystemStatus

ElevatorReport

- elevatorId: int
- currentFloor: int
- direction: Direction
- doorClosed: boolean
- floorRequests: boolean[]
- doorOpenTimer: int
- endWaitTimer: int
- outOfService: boolean
- takingRequests: boolean

+ isDoorClosed(): boolean
+ getElevatorId(): int
+ getDirection(): Direction
+ getEndWaitTimer(): int
+ isTakingRequests()
+ hashCode(): int
+ getFloorRequests(): boolean[]
+ getDoorOpenTimer(): int
+ isOutOfService(): boolean
+ getCurrentFloor(): int
+ toString(): String
+ equals(Object): bollean

«interface»
RequestInterface

+ getStartFloor(): int
+ getEndFloor(): int

Request(Class)

startFloor: int
endFloor: int

+Request(int, int)

+ getStartFloor(): int
+ getEndFloor(): int
+ toString(): String

buildingModelImpl(Class)

- numberOfFloors(int)
- numberOfElevators(int)
- elevatorCapacity(int)
- elevatorSystemStatus(enum)
- buildingReport(class)
- elevators (List<elevatorImpl>)
- requestList(List<Request>)

+ buildingModelImpl(int, int, int)

- initializeElevators(): List<elevatorImpl>
+ getElevatorSystemStatus(): elevatorSystemStatus
+ getNumberOfElevators(): int
+ getElevatorCapacity(): int
+ addRequest(List<Request>): boolean
+ distributeUpRequests(): void
+ startElevatorSystem(): boolean
+ stopElevatorSystem(): void
+ BuildingReport
+ step(): void
- clearRequestList(List<Request>): void

«enumeration»
elevatorSystemStatus

runniing
stopping
outOfService

- display: String

+ ElevatorSystemStatus(String)

+ toString: String

«interface»
buildingModel

+ getElevatorSystemStatus(): elevatorSystemStatus
+ getNumberOfElevators(): int
+ getElevatorCapacity(): int
+ addRequest(): boolean
+ distributeRequests(): void
+ startElevatorSystem(): boolean
+ stopElevatorSystem(): void
+ BuildingReport

«enumeration»
Direction

UP
DOWN
STOPPED

- display(String)

+ Direction(String)

+ toString(): String

elevatorImpl(Class)

- id: int
- maxFloor: int
- maxOccupancy: int
- currentFloor: int
- direction: Direction
- outOfService: boolean
- floorRequests: boolean[]
- takingRequests: boolean
- status: ElevatorStatus

+ ElevatorImpl(int, int)

+ getCurrentFloor(): int
+ getMaxFloor(): int
+ getMaxOccupancy(): int
+ getDirection(): Direction
+ getElevatorId(): int
+ getElevatorStatus: ElevatorStatus
+ isTakingRequests(): boolean
+ isDoorClosed(): boolean
+ getFloorRequests(): boolean[]
+ step(): void
- stepOutOfService(): void
- stepDoorOpen(): void
- stepTopOrBottom(): void
+ start(): void
+ processRequests(List<Request>): void
+ takeOutOfService(): void
- clearStopRequest(): void
+ toString(): String

«interface»
ElevatorInterface

+ step(): void
+ start():void
+ getElevatorId(): int
+ getMaxFloor(): int
+ getElevatorStatus(): ElevatorReport
+ processRequest(List<Request>): void
+ getCurrentFloor(): int
+ isTakingRequests(): boolean
+ takeOutOfService(): void
+ isDoorClosed(): boolean
+ getFloorRequests(): boolean[]
+ getDirection(): Direction
+ getMaxOccupancy(): int

«enumeration»
ElevatorStatus

running
stopping
stopped

- display: String

+ ElevatorStatus(String)

+ toString: String

Building Design

Xin Qi, CS5004 Final Project, 2024 Spring

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?

I am going to use an array list to store my elevators in my building model, each index of the array refers to a number(or id) for an elevator. Each elevator is an object in the array list, and the data structure for the elevators in the building model would be an array list of elevator objects.

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

I will use a list to store the incoming requests because the elements in a list are in order, which meets the demand of an elevator. I will try to implement a linked list for list distribution, because we need to let the first-in requests be first-out, and the time complexity of linked lists is better than other types of list like an arraylist.

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

Using the distributeRequests(List<Request>) method.

Find the elevators in the array list of the elevators in the building starting from the first index. For upRequests, if the current floor of the elevator is on the lowest floor, then distribute the request to this elevator. If not, check the next elevator in the array list. Once we find an elevator on the lowest floor, we will continue to distribute all the upRequests to it until it reaches its max capacity. After reaching its max capacity, we will check the following elevators. If there is no elevator on the lowest floor, wait until there is one.

For downRequests, if the current floor of the elevator is on the highest floor, then distribute the request to this elevator. If not, check the next elevator in the array list. Once we find an elevator on the highest floor, we will continue to distribute all the downRequests to it until it reaches its max capacity. After reaching its max capacity, we will check the following elevators. If there is no elevator on the highest floor, wait until there is one.

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

We could have a private method to clear all the stop requests for each elevator.
For each elevator, check if the elevator is on the lowest or highest floor.

- If the elevator is on these floors:
    - change the status for taking request to false
    - change the status of door to open

        - change the status of elevator to stopped
- If the elevator is not on the lowest or highest floors
    - immediately process a request to the lowest floor, make the direction of the elevator down
        - change the status for taking request to false
        - When reaching the ground floor, change the door to open
        - Change the status of elevator to stopped

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

The step method in the buildingModelImpl updates the state of each elevator based on its current state(running or stopping) and pending requests, including moving the elevators up and down, opening or closing doors(keep the door open for steps), and servicing requests (go to the top or bottom after waiting for 5 steps). We could use 3 private methods, stepOutOfService, stepDoorOpen, stepTopOrBottom for the public step method.

6) How do you start processing requests?

The startElevatorSystem method of the buildingModelImpl class initiates the processing of requests by starting all elevators.
Elevator system status changes to running.
Requests are processed by starting the elevator system.
Once the system is running, requests can be accepted(addRequest could return true and requests could be added to the request list), all the elevator status changes from stopped to running and the requests are assigned to elevators based on their capacity.

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

The stopElevatorSystem method of the buildingModelImpl class takes the building out of service.
All new requests are rejected (addRequest returns false and does not add requests to the request list).
Clear all existing requests in the request list using List.clear()method.
Set every elevator’s elevatorStatus in the building to “stopped”.
Set the elevatorSystemStatus to outOfService.

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

Each elevator has a takeOutOfService method which puts it out of service. Calling the methods for all elevators in the building. When calling this method:
Clear all existing requests in the request list using clearRequestList(List<Request>) method in the buildingModelImpl.
Check the status of each elevator in the list of elevators using getElevatorStatus().

- If the status of the elevator is running, wait until its status becomes stopping. After that, let the stopping elevator process a new request to the lowest floor. When the elevator reaches the lowest floor and is stopping, change its status to stopped.
- If the status of the elevator is stopping, check if it is on the lowest floor using getCurrentFloor().
    - If it is, change the status to “stopped”.

    - If it isn’t, let the stopping elevator process a new request to the lowest floor. When the elevator reaches the lowest floor and is stopping, change its status to stopped.