<u>USE CASES</u>

USE CASE: USE ELEVATOR
Primary Actor: Elevator Passenger
Scope: Elevator
Level: User Goal
Precondition: A passenger requires an elevator to traverse floors
Postcondition: Elevator arrives at floor
Trigger: A passenger decides to use an elevator
Minimal Guarantee: Arrival to a safe floor
Success Guarantee: Arrival to desired floor
Main Success Scenario:

1. A button, marked either “up” or “down” is pressed on one of the N floors
    a. The button illuminates and stays illuminated
2. One of the M elevators arrives to the floor where the button was pressed
    a. The button light turns off
    b. Elevator rings a bell
    c. The elevator and floor doors open for a fixed amount of time of 10 seconds
3. Passenger enters the elevator
4. Elevator bell rings again
5. Elevator door closes
6. The passenger chooses one or more destination floors, or another button, from a panel of buttons
7. Passenger traverses the floors
8. Elevator display updates after each floor the elevator passes
9. Elevator sensor notifies the elevator when it has reached its desired floor
10. Elevator arrives to the desired floor
    a. Bell rings
    b. Elevator and floor doors open for a fixed time of 10 seconds
11. Passenger exits the elevator

Variations:

6a. Passenger holds “open” door button
    6a1. The door stays open relative to the amount of time the passenger is holding the button for.
6b. Passenger holds the “close” door button
    6b1. The door closes prematurely.

USE CASE: HELP SIGNAL
Primary Actor: Elevator Passenger
Scope: Elevator
Level: subfunction
Precondition: The help button has been pressed inside the elevator
Success Guarantee: Help is provided
Main Success Scenario:

1. Passenger is connected to building safety service through a voice connection

Extensions:

1a. No response from building safety within 5 seconds
    1a1. A 911 emergency call is placed
1b No response from passenger within 5 seconds
    1b1. A 911 emergency call is placed

USE CASE: DOOR OBSTACLE
Primary Actor: Elevator passenger
Scope: Elevator
Level: subfunction
Precondition: The light sensor is interrupted when the door is closing
Postcondition: Elevator door open
Trigger: An object between the doors
Success Guarantee: Elevator door will stop from closing and be opened
Main Success Scenario:

1. The control system stops the door from closing and opens it

Extensions:

1a. Door obstacle occurs repeatedly over a short period of time
    1a1. A warning is sounded over the audio system
    1a2. A text message is displayed

USE CASE: FIRE SIGNAL
Primary Actor: Elevator Passenger
Scope: Elevator
Level: subfunction
Precondition: A fire detected
Trigger: A fire erupts
Success Guarantee: Elevator arrives to a safe floor
Main Success Scenario:

1. Elevator receives a “Fire” alarm signal
2. Elevator traverses to a safe floor
3. an audio and text message are presented to passengers informing them of an emergency and asking them to disembark once the safe floor is reached

USE CASE: OVERLOAD SIGNAL
Primary Actor: Elevator Passenger
Scope: Elevator
Level: subfunction
Precondition: Weight on elevator is higher than what is allowed
Trigger: Weight on elevator is exceeded
Minimal Guarantee: Passengers are notified of the weight problem
Success Guarantee: Weight is reduced and the elevator can move again
Main Success Scenario:

1. The elevator does not move
2. audio and text messages are presented to passengers asking for the load to be reduced before attempting to move again.

USE CASE: POWER OUT SIGNAL
Primary Actor: Elevator Passenger
Scope: Elevator
Level: subfunction
Precondition: Power is out within building/elevators
Trigger: Power goes out
Success Guarantee: Elevator arrives to a safe floor
Main Success Scenario:

1. The control system receives a “Power Out” alarm signal
2. An audio and a text messages are presented to passengers informing them of the power outage
3. Each elevator is then moved to a safe floor
4. passengers are asked to disembark via audio and text messages

TRACEABILITY MATRIX

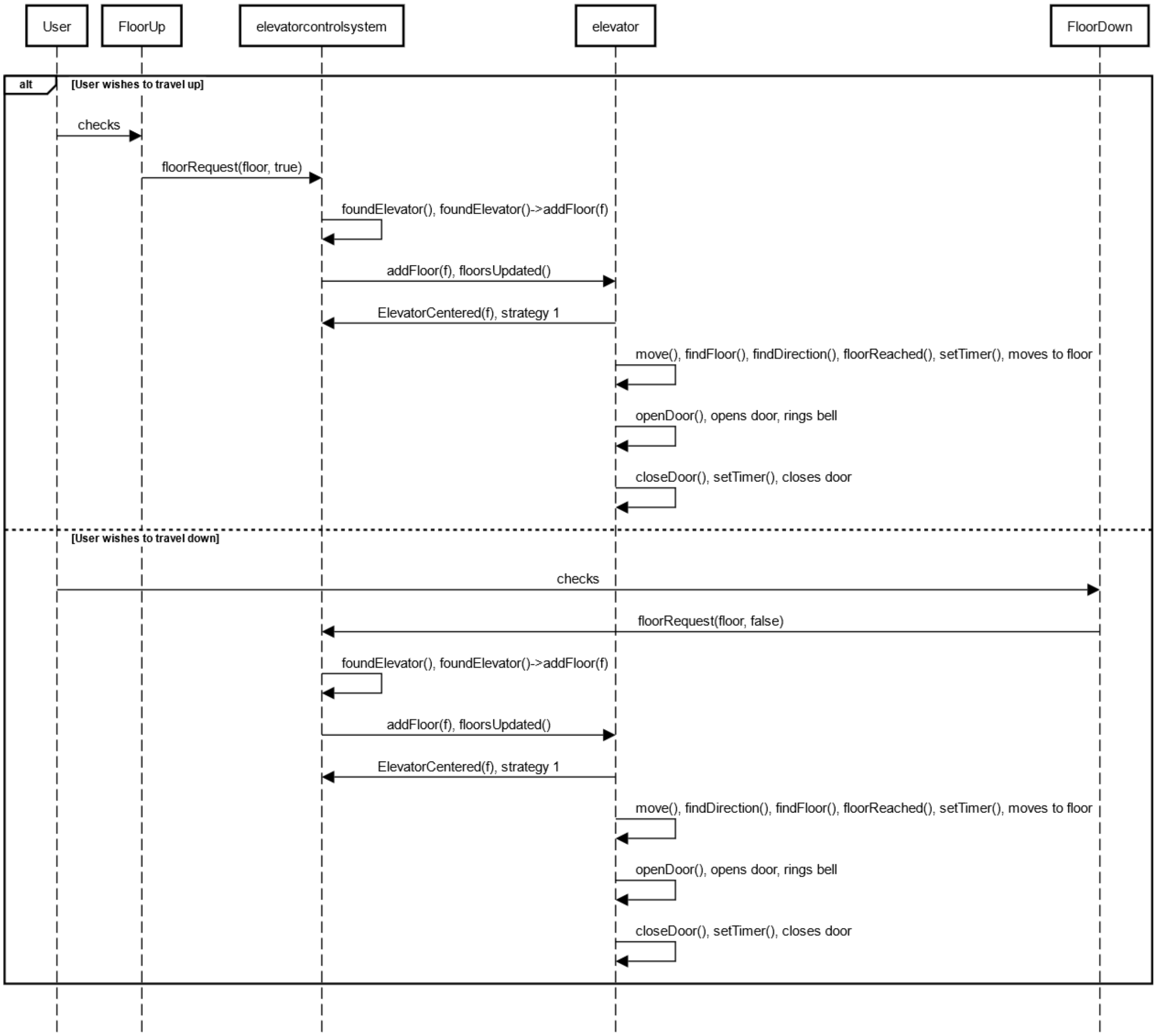
ID	REQUIREMENT	RELATED USE CASE	FULFILLED BY	IMPLEMENTED BY	TESTED BY
1	Buttons marked either up/down at floors, panel in elevator, close door and open door, bell rings	USE ELEVATOR	elevator, elevatorcontrolsystem, floor	simulateCases(), closeDoor(), openDoor(), setTimer()	Using the “Floor Up” and “Floor Down” buttons, using the checkboxes in the elevators panel. (outputted in console, also seen in individual “Floor #” checkboxes updating
2	Sensor which is used to notify when elevator has arrived at a floor	USE ELEVATOR	elevator	findFloor(), floorReached(), inFloors(Floor)	Using elevators panel and sending elevator to certain floors, checking if they reached the proper floor (outputted in console)
3	Display and audio system to handle messages, and connections, to (and from) the passenger	(ALL USE CASES)	elevator, elevatorcontrolsystem, floor	Individual implementation across multiple functions	Triggering events that lead to audio system and display functionality (i.e floor traversal, “Help” check box, …)

					(outputted in console)
4	“Help” button inside the elevator, sends “Help” signal and connects passengers to building safety, if unsuccessful, call 911	HELP SIGNAL	elevator, elevatorcontrolsystem, floor	helpCase(int), simulateCases(), handleSimCase(), handleCaseOnClose()	Checking off each elevators “Help” check box (outputted in console)
5	Light sensor detects obstacles in between the doors, informs passengers	DOOR OBSTACLE	elevator	obstacleCase(int), simulateCases(), handleSimCase(), openDoor(), closeDoor(), handleCaseOnClose(), setTimer()	Checking off “Door Obstacle” checkboxes for each elevator (outputted in console)
6	Fire is detected and a “Fire” signal is sent, in turn moving elevators to a safe floor and notifying passengers	FIRE SIGNAL	elevator, elevatorcontrolsystem, floor	fireCase(int), simulateCases(), handleSimCase(), openDoor(), closeDoor(), handleCaseOnClose(), setTimer()	Checking off “Fire in Building” checkbox or checking off individual elevators “Fire” check box (outputted in console)
7	Overload in elevator, “Overload” signal is sent, passengers are told to reduce weight, elevator does not move	OVERLOAD SIGNAL	elevator,	overloadCase(int), simulateCases(), handleSimCase, openDoor(), closeDoor(), handleCaseOnClose(), setTimer()	Checking off “Overload” checkboxes for each elevator (outputted in console)
8	Power is out, “Power Out” signal is sent, all elevators moved to a safe floor, passengers notified	POWER OUT SIGNAL	elevator, elevatorcontrolsystem, floor	poweroutCase(int), simulateCases(), handleSimCase(), openDoor(), closeDoor(), handleCaseOnCl	Checking off “Power Out” checkbox (outputted in console)

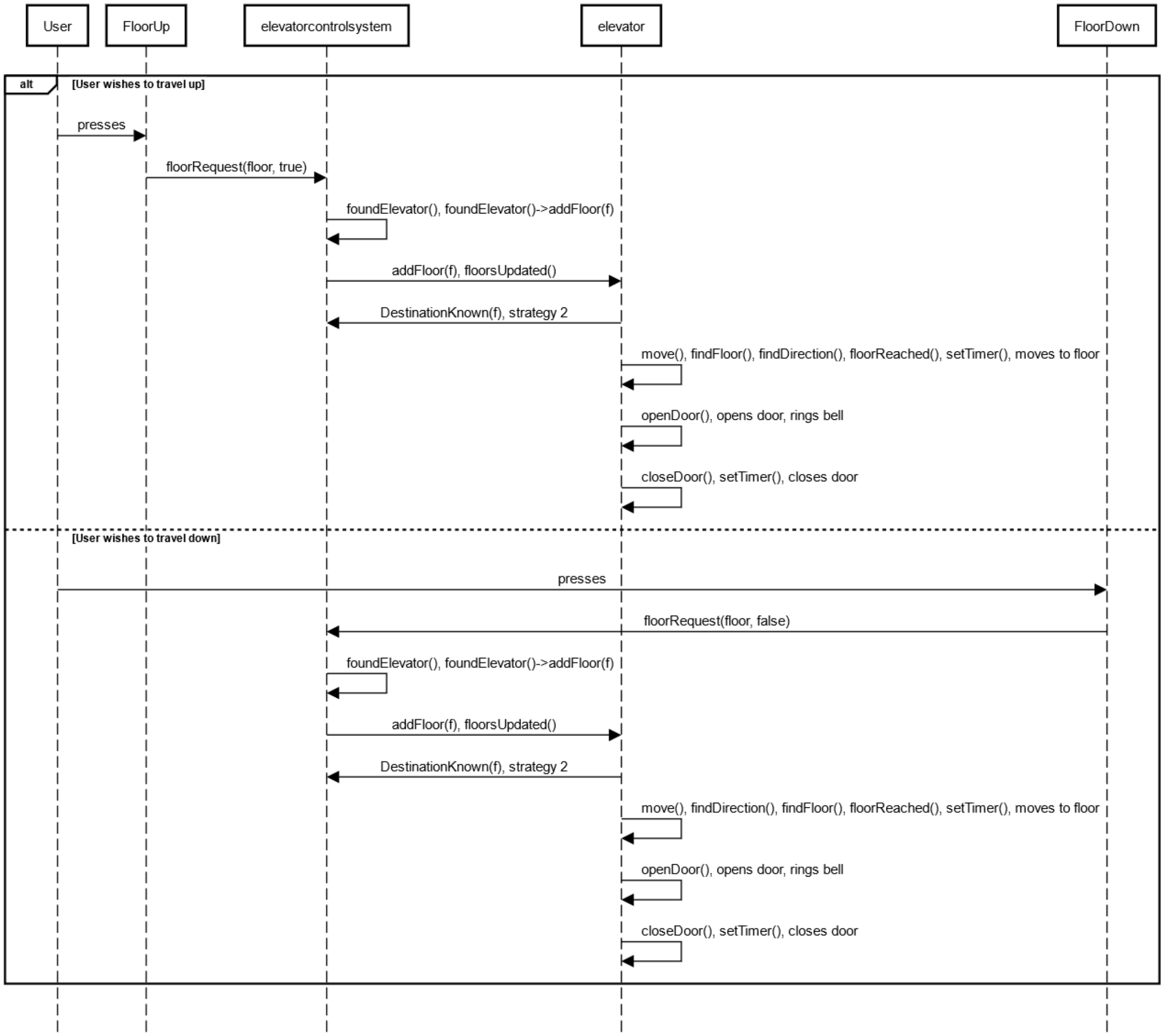
				ose(), setTimer()	

SEQUENCE DIAGRAMS

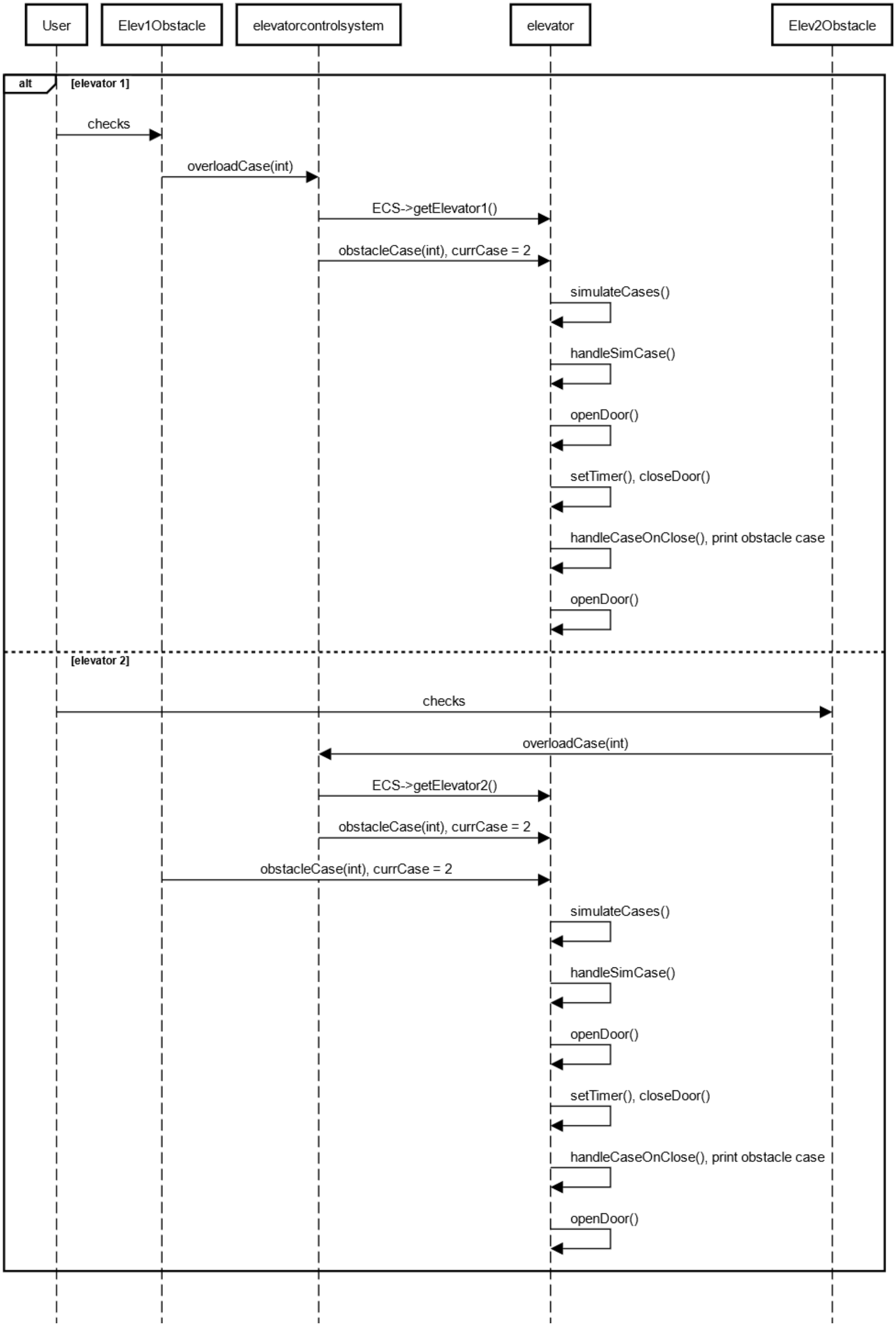
BASE Use Case 1


User
FloorUp
elevatorcontrolsystem
elevator
FloorDown
alt
[User wishes to travel up]
checks
floorRequest(floor, true)
foundElevator(), foundElevator()->addFloor(f)
addFloor(f), floorsUpdated()
ElevatorCentered(f), strategy 1
move(), findFloor(), findDirection(), floorReached(), setTimer(), moves to floor
openDoor(), opens door, rings bell
closeDoor(), setTimer(), closes door
[User wishes to travel down]
checks
floorRequest(floor, false)
foundElevator(), foundElevator()->addFloor(f)
addFloor(f), floorsUpdated()
ElevatorCentered(f), strategy 1
move(), findDirection(), findFloor(), floorReached(), setTimer(), moves to floor
openDoor(), opens door, rings bell
closeDoor(), setTimer(), closes door

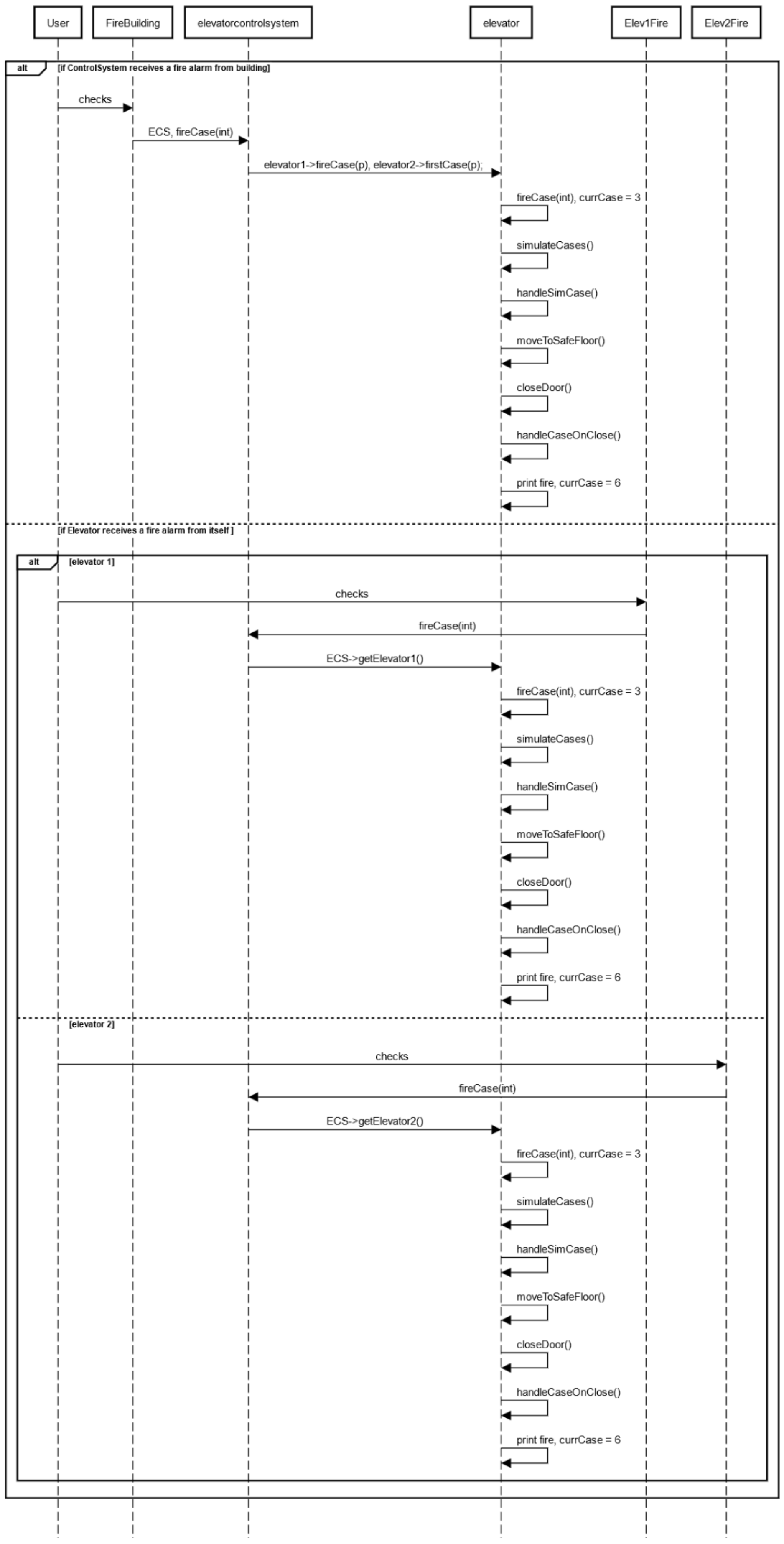
BASE Use Case 2


User
FloorUp
elevatorcontrolsystem
elevator
FloorDown
alt
[User wishes to travel up]
presses
floorRequest(floor, true)
foundElevator(), foundElevator()->addFloor(f)
addFloor(f), floorsUpdated()
DestinationKnown(f), strategy 2
move(), findFloor(), findDirection(), floorReached(), setTimer(), moves to floor
openDoor(), opens door, rings bell
closeDoor(), setTimer(), closes door
[User wishes to travel down]
presses
floorRequest(floor, false)
foundElevator(), foundElevator()->addFloor(f)
addFloor(f), floorsUpdated()
DestinationKnown(f), strategy 2
move(), findDirection(), findFloor(), floorReached(), setTimer(), moves to floor
openDoor(), opens door, rings bell
closeDoor(), setTimer(), closes door

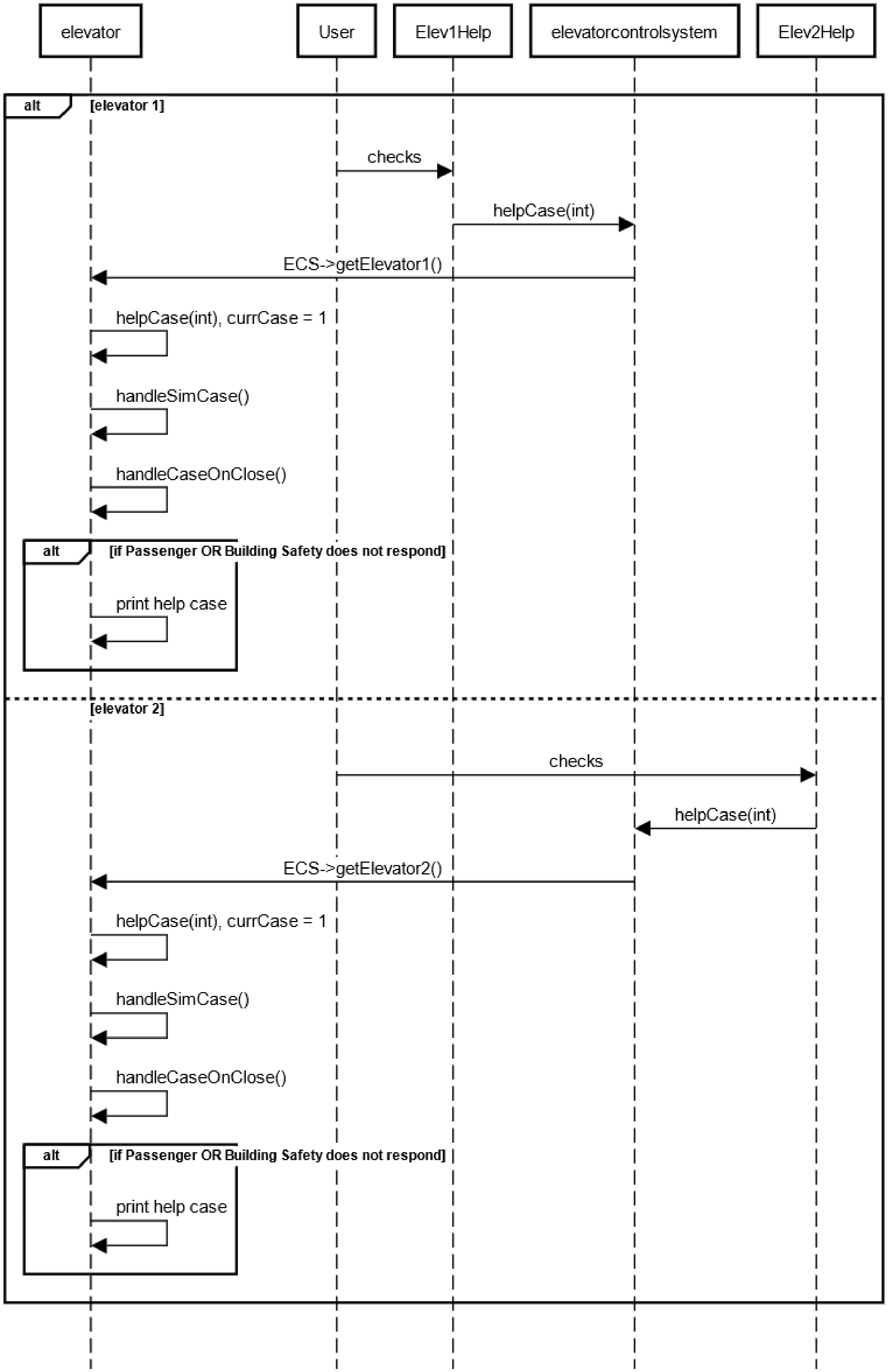
DOOR OBSTACLE Use Case



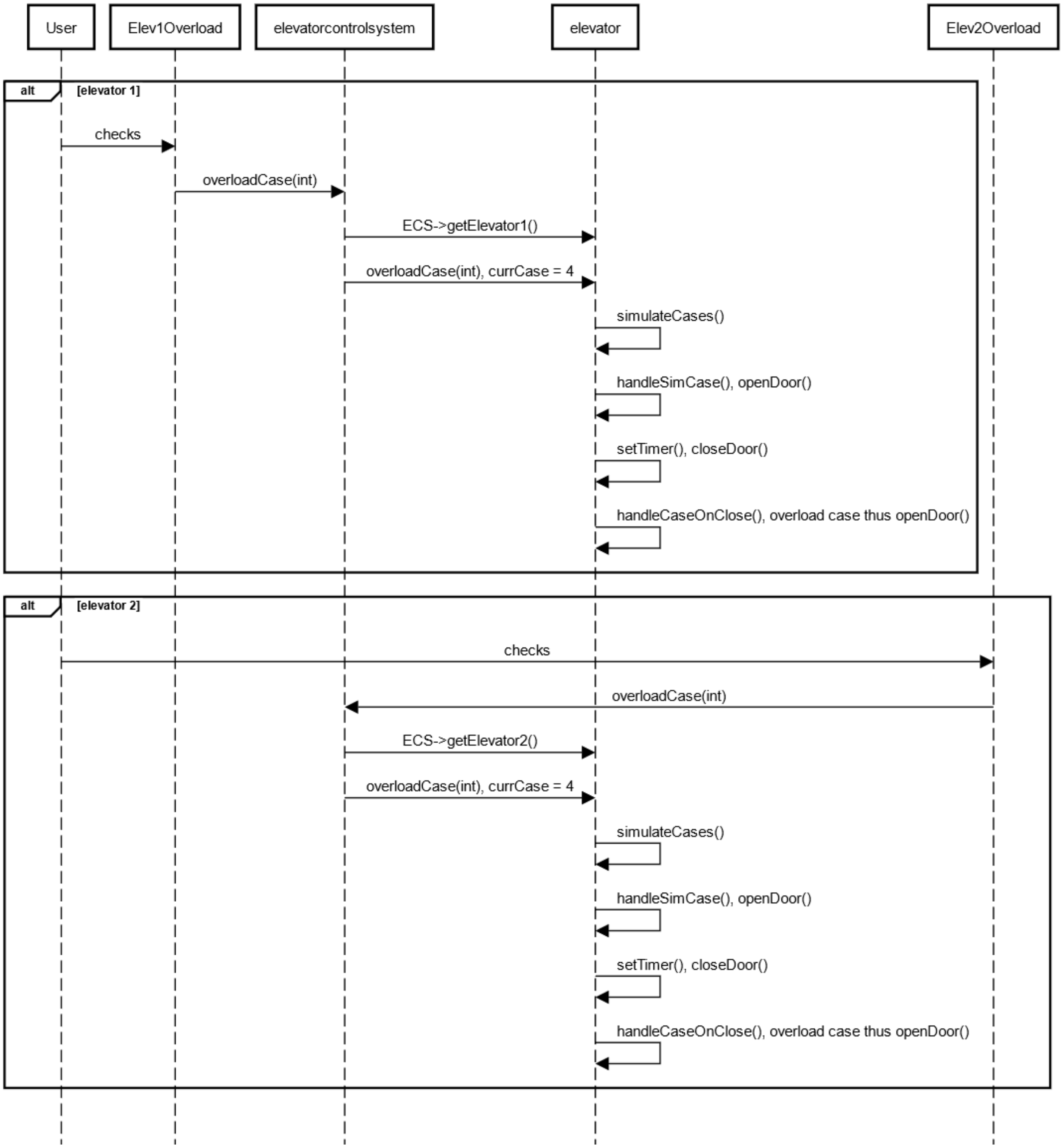
FIRE Use Case



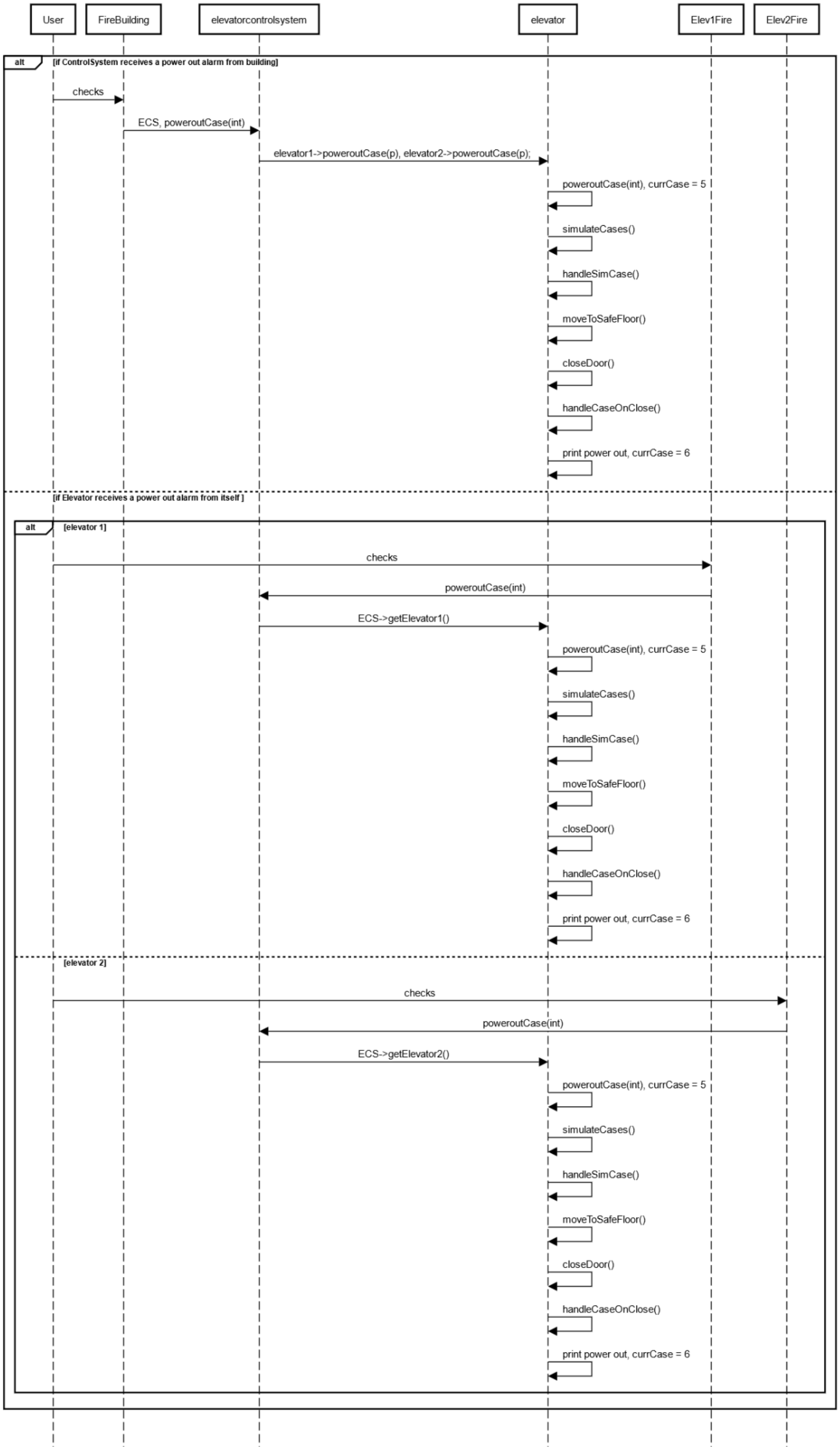
HELP Use Case



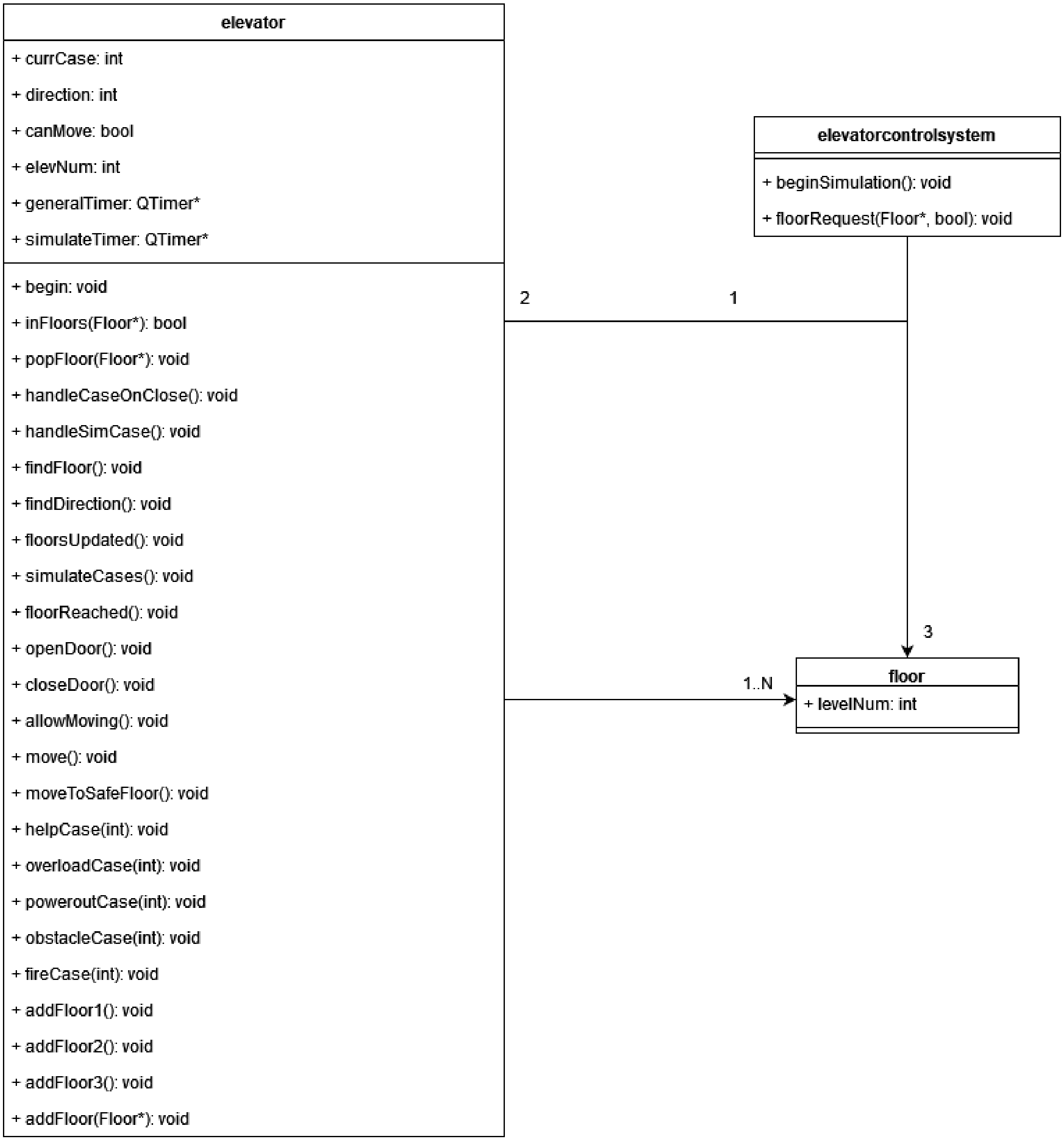
OVERLOAD Use Case



POWER OUT Use Case



UML DIAGRAM


elevator
+ currCase: int
+ direction: int
+ canMove: bool
+ elevNum: int
+ generalTimer: QTimer*
+ simulateTimer: QTimer*
+ begin: void
+ inFloors(Floor*): bool
+ popFloor(Floor*): void
+ handleCaseOnClose(): void
+ handleSimCase(): void
+ findFloor(): void
+ findDirection(): void
+ floorsUpdated(): void
+ simulateCases(): void
+ floorReached(): void
+ openDoor(): void
+ closeDoor(): void
+ allowMoving(): void
+ move(): void
+ moveToSafeFloor(): void
+ helpCase(int): void
+ overloadCase(int): void
+ poweroutCase(int): void
+ obstacleCase(int): void
+ fireCase(int): void
+ addFloor1(): void
+ addFloor2(): void
+ addFloor3(): void
+ addFloor(Floor*): void
elevatorcontrolsystem
+ beginSimulation(): void
+ floorRequest(Floor*, bool): void
2
1
3
1..N
floor
+ levelNum: int

Textual Explanation for Design Decisions

For my design I decided to go with 3 main classes representing the elevator, the control system, and floors. I also decided to use QTimer as a main part of my design, controlling the main simulation (along with other uses such as closing and opening doors for instance). For the purposes of this assignment I included 3 floors and 2 elevators for seamless testing. The GUI is used to represent the passengers' view in different stages of the system, along with console output for various tests, events, and the main scenario. Apart from that there are views that are used specifically for testing specific events such as for instance overload, fires, or power outs. As for strategies, I decided to use the destination known strategy as well as an elevator centered strategy. When triggering certain events, some checkboxes will be greyed out and not available for use, as during some events the program will not allow specific boxes and buttons to be pressed or checked (i.e during a fire in building event, user will not be able to select a floor to travel to as the elevator will travel to a safe floor on its own). Finally, for this program there is a set safe floor which is always floor 1.

Instructions for Program Use and Testing

To simulate the main scenario, the user can choose a floor to start from by choosing a floor in the combo box found in the Floor view, once a floor is selected the user can request an elevator that travels up or down using the “Floor Up” and “Floor Down” buttons.

Once an elevator has arrived (seen in console output) the user can choose a floor, choose help, or choose to open and close the door, in the respective elevator's elevator view, represented by the “Floor 3”, “Floor 2”, “Floor 1”, “Help”, “Open Door”, and “Close Door” checkboxes. The elevator travelling and reaching the floor, help, open and close door, can be seen in the console output, as well as individual floor checkboxes being checked off when reaching that floor.

Each elevator also has their own testing area, in which the user can choose to test certain events that act upon each individual elevator, once again outputted in console. In this program there are 3 checkboxes for each elevator, the events overload, fire in a certain elevator, and obstacle in door, are tested respectively by the “Overload”, “Fire Elevator” and “Obstacle in Door” checkboxes, and outputted in console.

There is a general testing area which is used for testing events that affect both elevators, in this program there are 2 checkboxes, “Power Out” and “Fire in Building”, that test power out and fire in building events, once again outputted in console.

There is also a button labeled “CHECK CURRENT FLOORS” which outputs the current floors for each elevator, for easier testing, outputted in the console.

Finally, the user is provided an area to switch between strategies 1 and 2, the switch is outputted in the console.

NOTE: upon using any of the buttons or checkboxes, or doing anything in fact, give the program some time to perform the action and output in console as I had to increase timer times as to avoid duplicate output in certain scenarios, also the program was built in **qmake**.