WASD Team:

Ramilya Yusupova (Backend Developer)

Arina Smirnova (Project Manager)

Sergey Zhurbey (Backend Developer, Tester)

Andrey Kan (Frontend Developer)

Mahmoud Hossameldin (Frontend Developer)

Task 2. Sequence diagrams

a) After the pressButton operation, the closeDoors operation is returned as a result of synchronous operation. Then, the startMoving is invoked synchronously since the lift cannot perform any other synchronous operations at that moment. The lift only sends asynchronously floorReached operation with the number of the floor passed as an argument. After the stopMoving operation is called, the user calls the pressDoors operation to open the doors with the openDoors operation as a feedback from the lift.
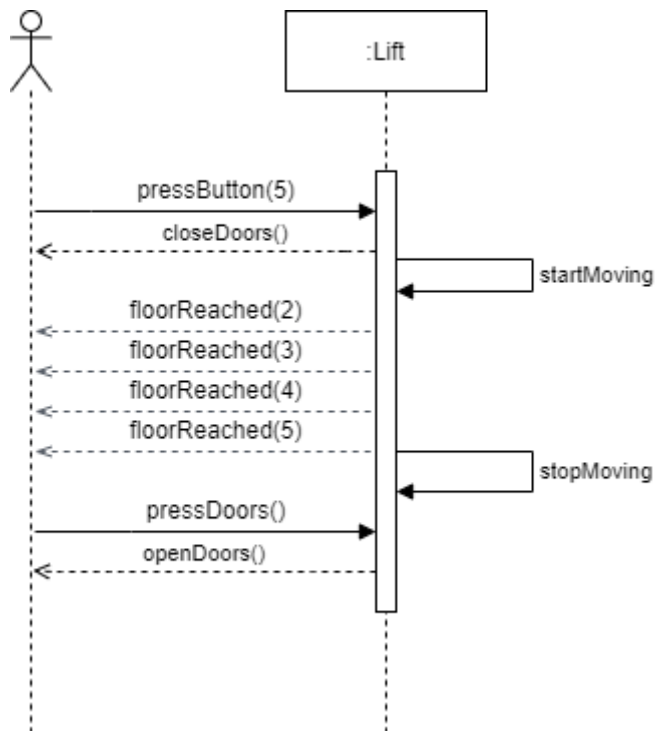

b) This diagram represents pretty the same actions order as under a), except that the first operation, that is being sent to Lift is pressDoors. It sends a corresponding answers – closeDoors. After that, User calls pressButton with argument '5', which relates to the desired floor number, and Lift starts moving.
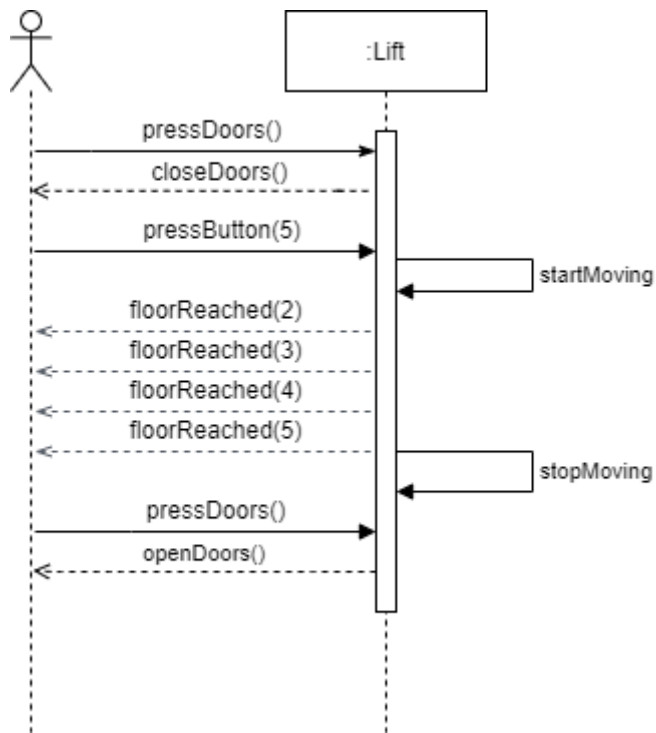
c) To signalize that User can not call for opening or closing doors while Lift is moving, the Opt frame is used. If User presses the button, call pressDoors is being sent, but no response from Lift comes after that. If the Lift has already made an action of stopMoving, calling a pressDoor action will get openDoor response.
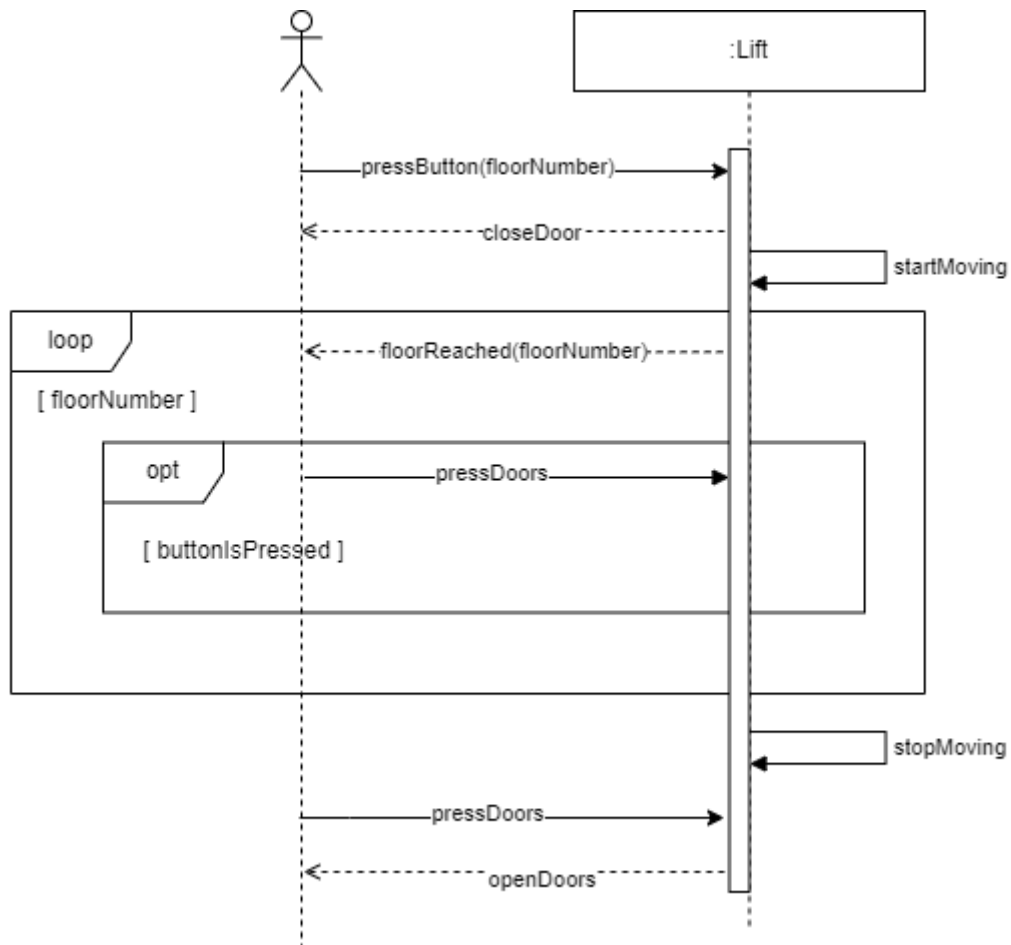
d) Ones the button has been pressed, lift closes the door and starts moving. To represent that the lift spends exactly 3 seconds reaching each floor we used a time constraint element. On the diagram after startMoving method invocation goes a loop, each iteration of which ends with floorReached action. From the start of iteration and its completion passes 3 seconds.
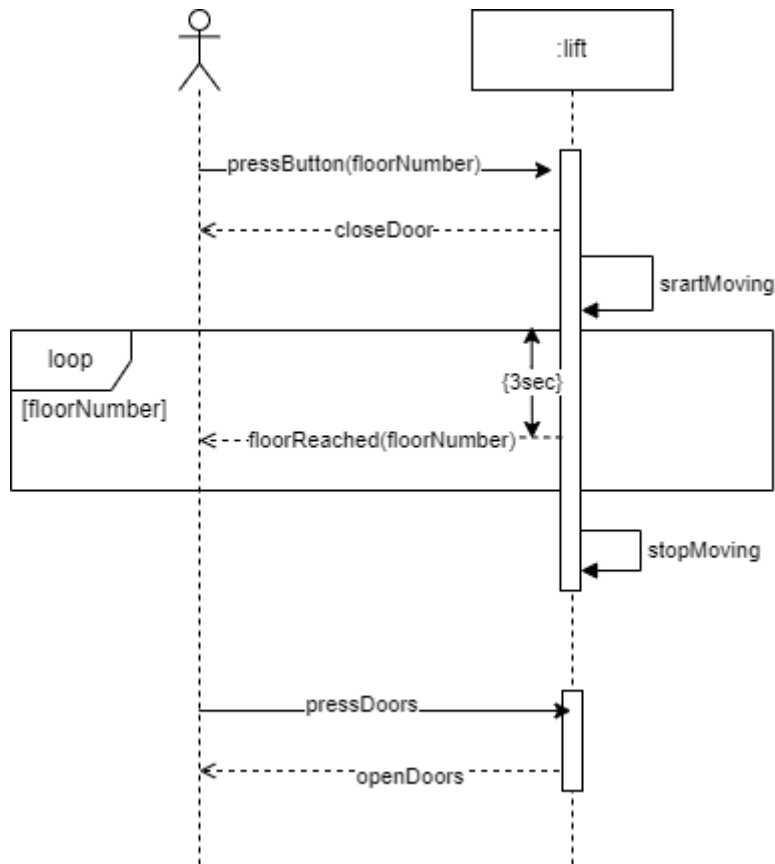
e) Class diagram contains three classes, one of which (Lift) is active. That means it manages its own execution.

Passenger class is sending a signal pressButton, which is received and handled

by Lift class (it has a reception for signal pressButton). This signal has one attribute,

which has a constraint, floorNumber can't be less than minimum Lift floor number,

or bigger than maximum floor number.

Display class has a composition relation to Lift class, which means it depends on it's

Life cycle and dies when owner class is getting destroyed.

<<signal>>
pressButton

+ floorNumber: int

{Lift.bottomFloorNumber
<= floorNumber
<= topFloorNumber}

<<class>>
Passanger

+ weight: real

passangersInside
0..N

choosenElevator
1

<<class>>
Lift

+ topFloorNumber: int
+ bottomFloorNumber: int
+ currentFloor: int

+ pressButton(floor: int)
+ pressDoors()
# startMoving(floor: int)
# stopMoving()
# <<signal>> pressButton

1

1

<<class>>
Display

+ showCurrentFloor()

max number of passengers N depends on
the total weight of people inside or the volume
they occupy