Task - 7

Team: ETL-Express

Description:

Solve behavior modeling tasks using UML2. This is a research and self-study activity. You will need to find out about new concepts using the references and links provided.

You will get individual points for studying and explaining new concepts to other students. Prepare a short 5-7 minutes report on the topic. A report shall include a correct solution of the bullet from the task as an example.

The most elaborate and precise report for each new concept gets the point.

A team gets a project point if it correctly solves all of the tasks and bullets in a chosen category.

There are a total of eight new concepts in four modeling tasks (one in a bullet). There are three categories. No more than four teams per category. You may coordinate concepts and categories with the other teams. Please, at most two reports on the same concept.

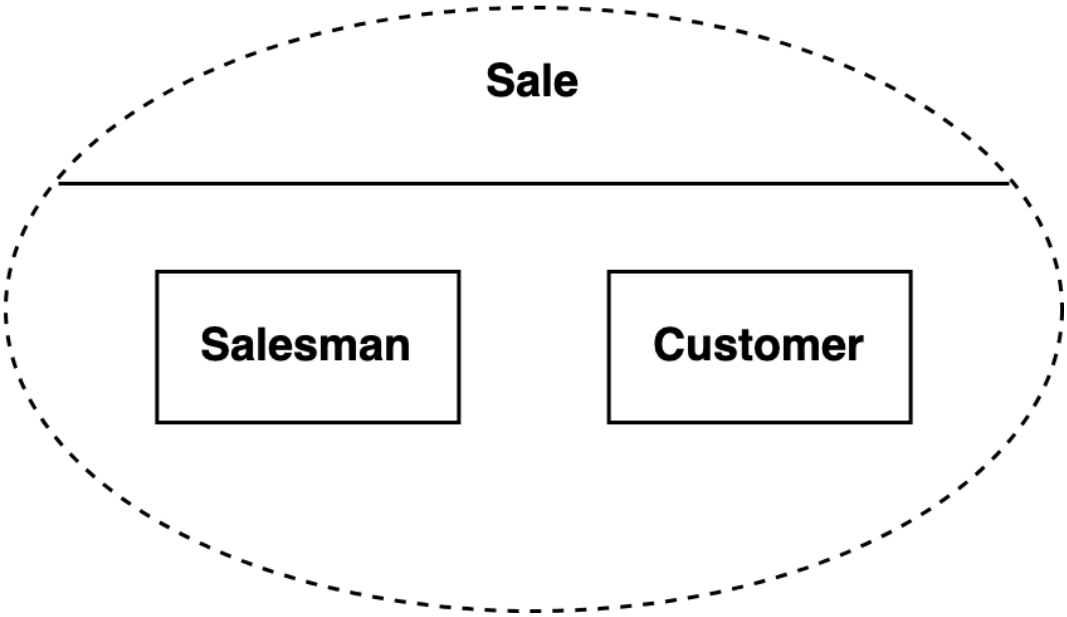
There is an extra Task 3 for activity modeling. Solve it at will.

Results of the task include

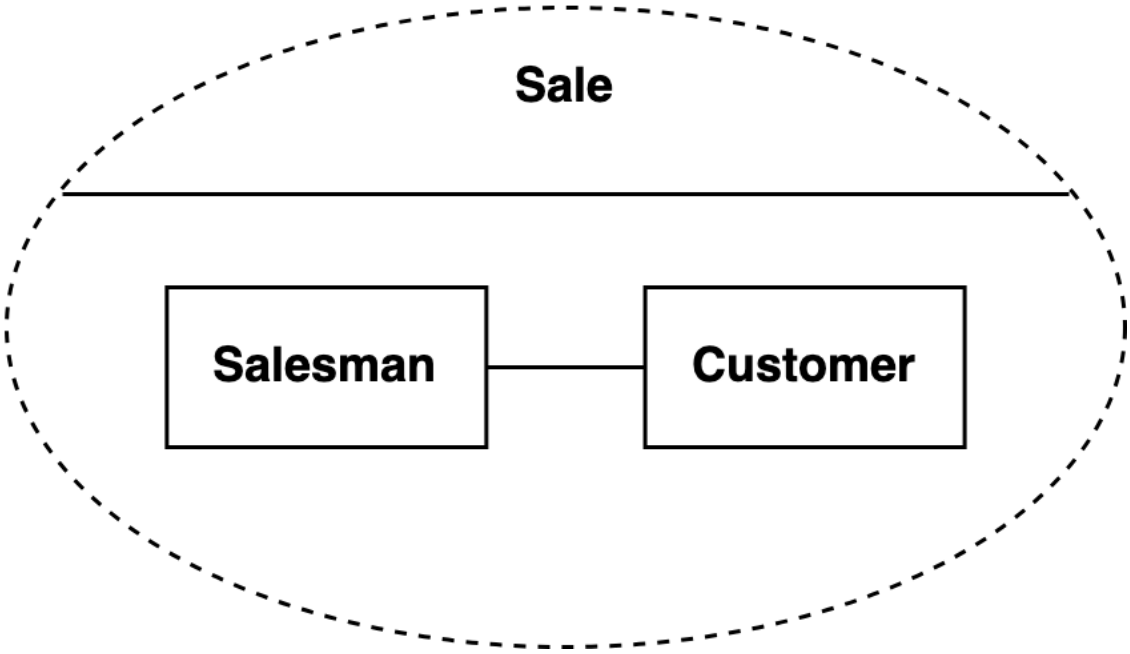
- a PDF with solved tasks in the chosen category and rationale for the solution
- a video recording with an explanation of a new concept (if not reported at class)

Task - 1

TASK 1 - BASE



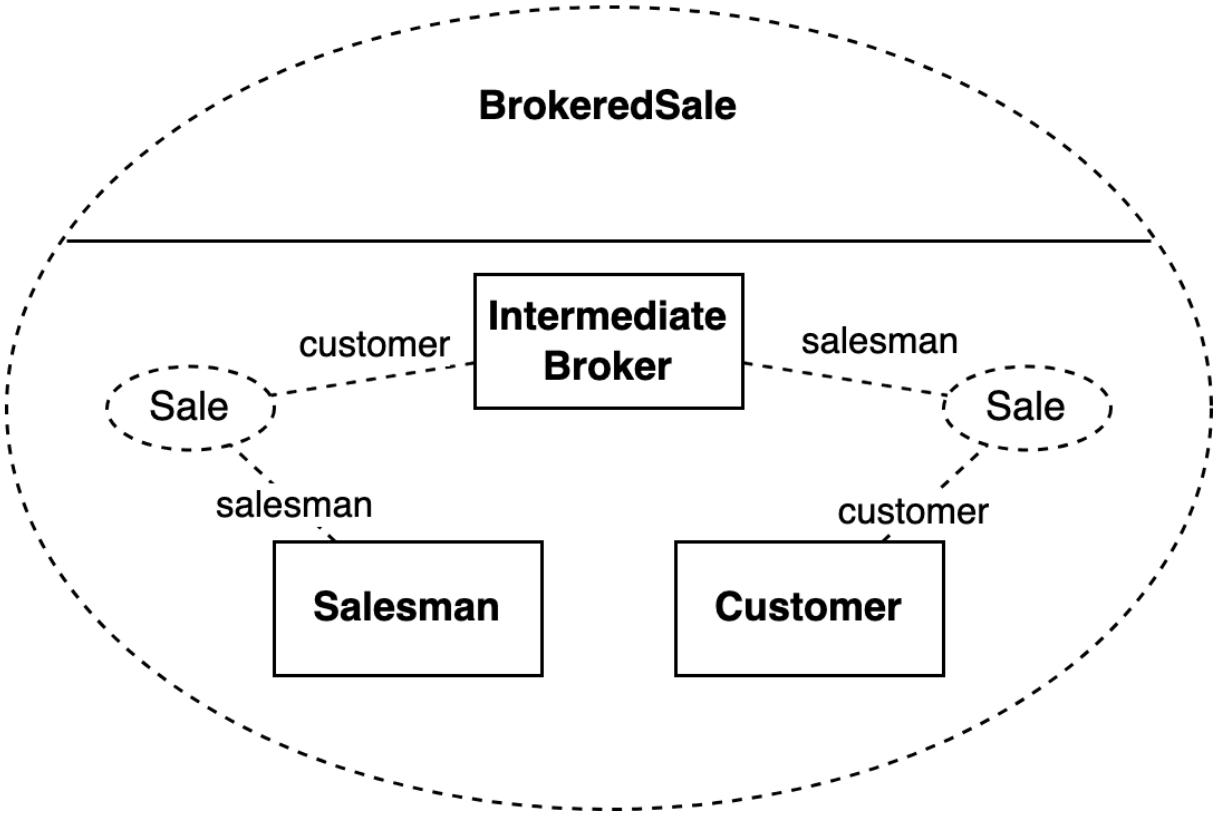
TASK 1 - A



Description:

Added an association between two classes Salesman and Customer

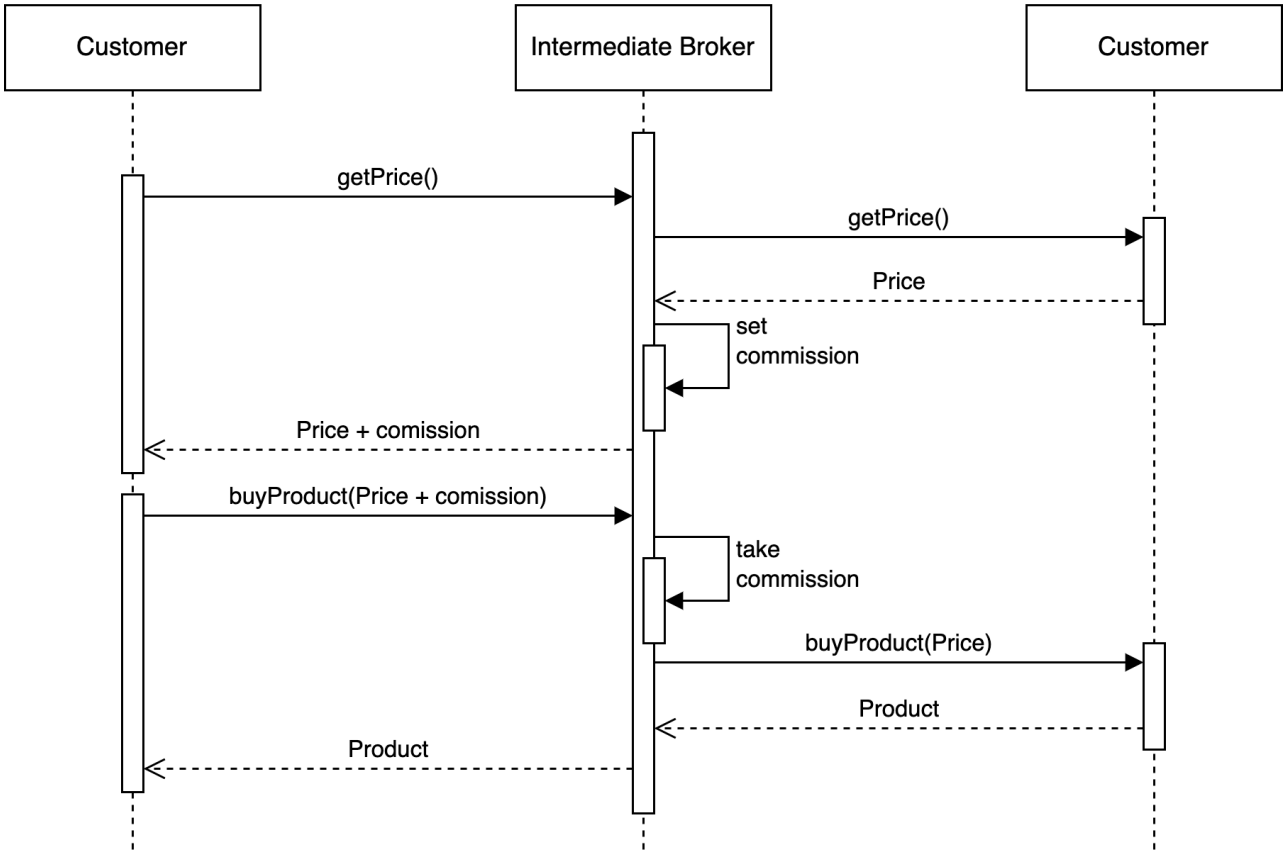
TASK 1 - B



Description:

Created a new cooperation, where the Intermediate Broker is connected to Salesman and Customer via the Sale cooperation from point A. When connected to Salesman, the Intermediate Broker acts as Customer, and when connected to Customer, it acts as Salesman.

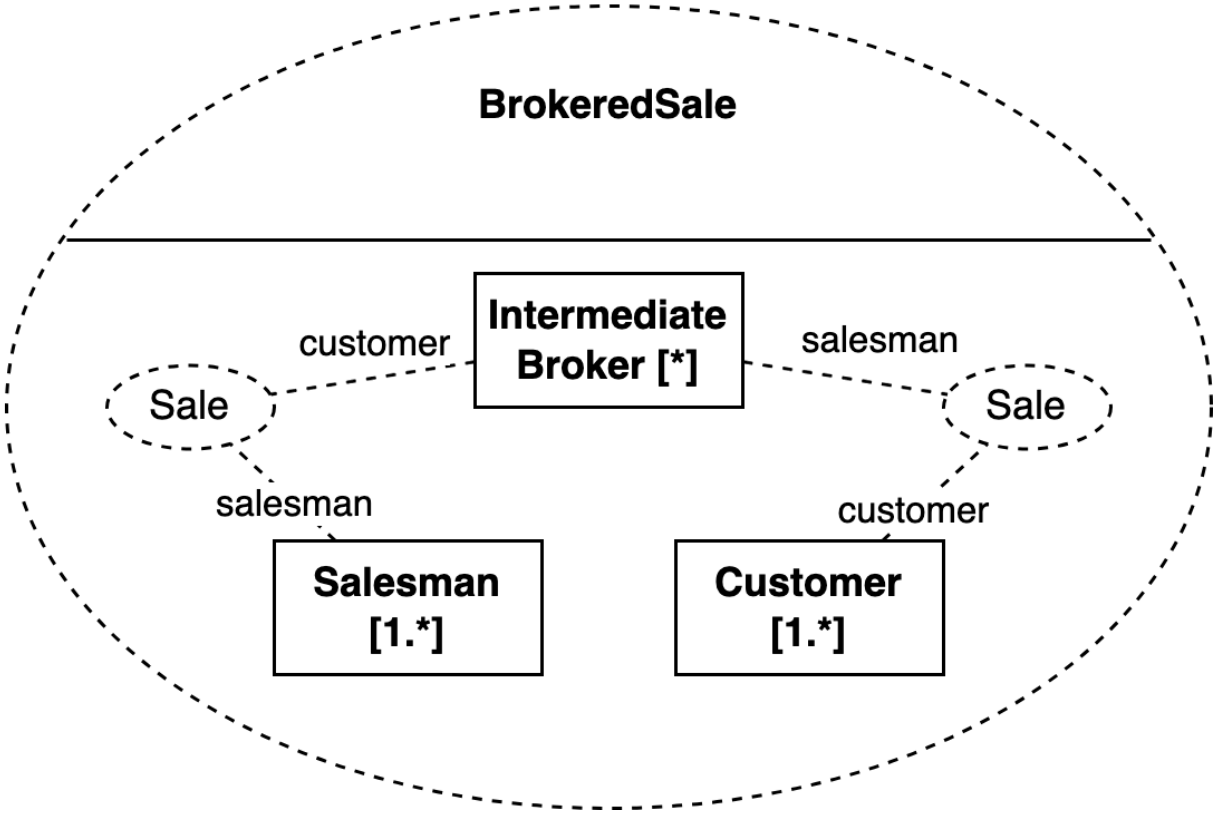
TASK 1 - C



Description:

A visualization of cooperation is presented in the form of a sequence diagram.

TASK 1 - D

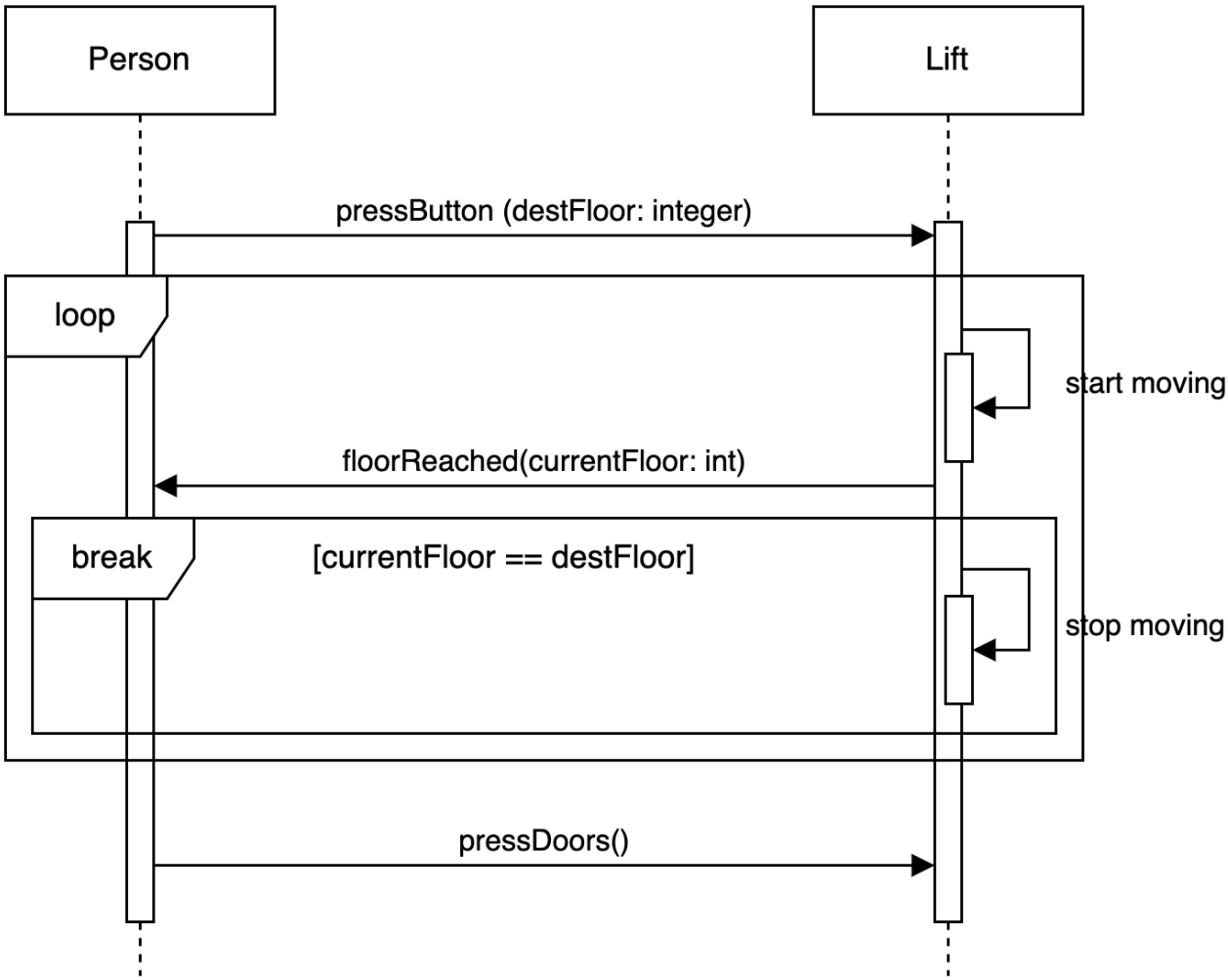


Description:

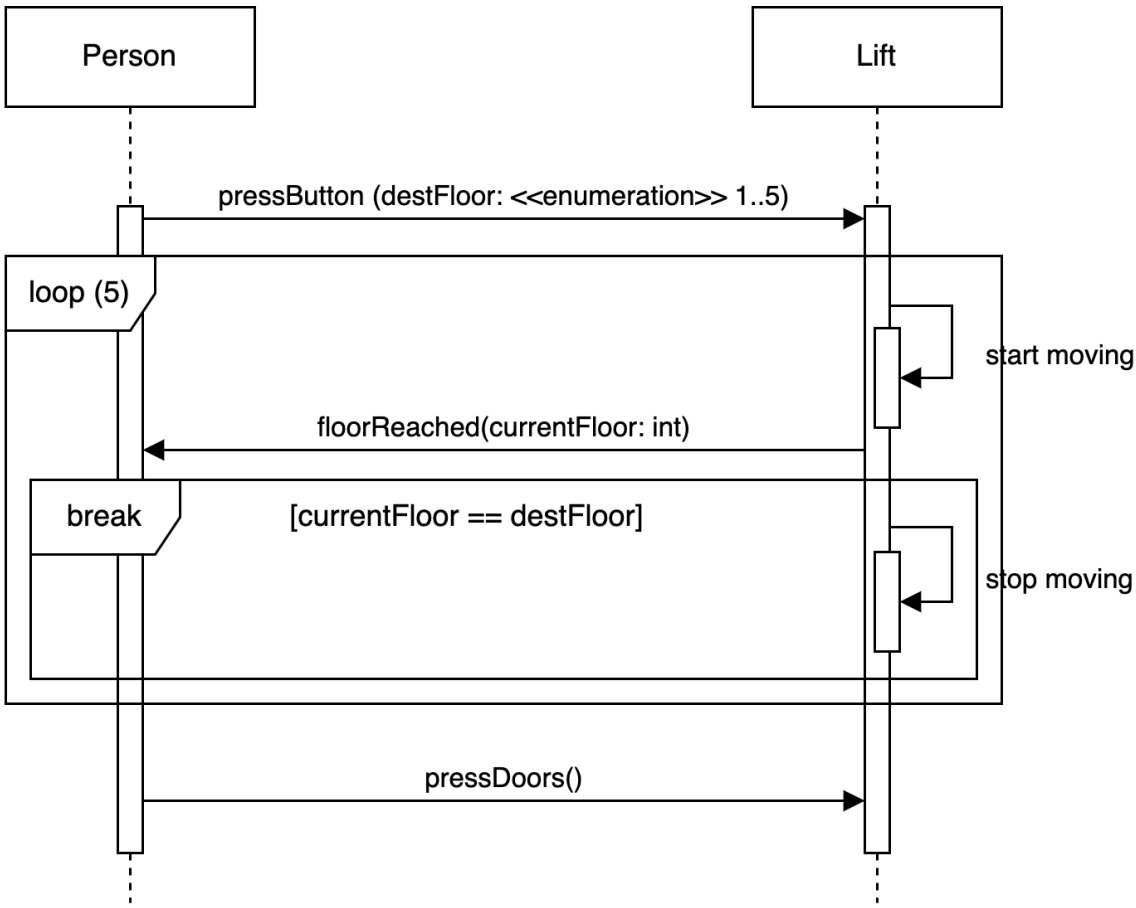
In the cooperation diagram from point B, we add an indication that there can be many brokers and they interact with several salesmen and customers.

Task - 2

TASK 2 - BASE


Person
Lift
pressButton (destFloor: integer)
loop
start moving
floorReached(currentFloor: int)
break
[currentFloor == destFloor]
stop moving
pressDoors()

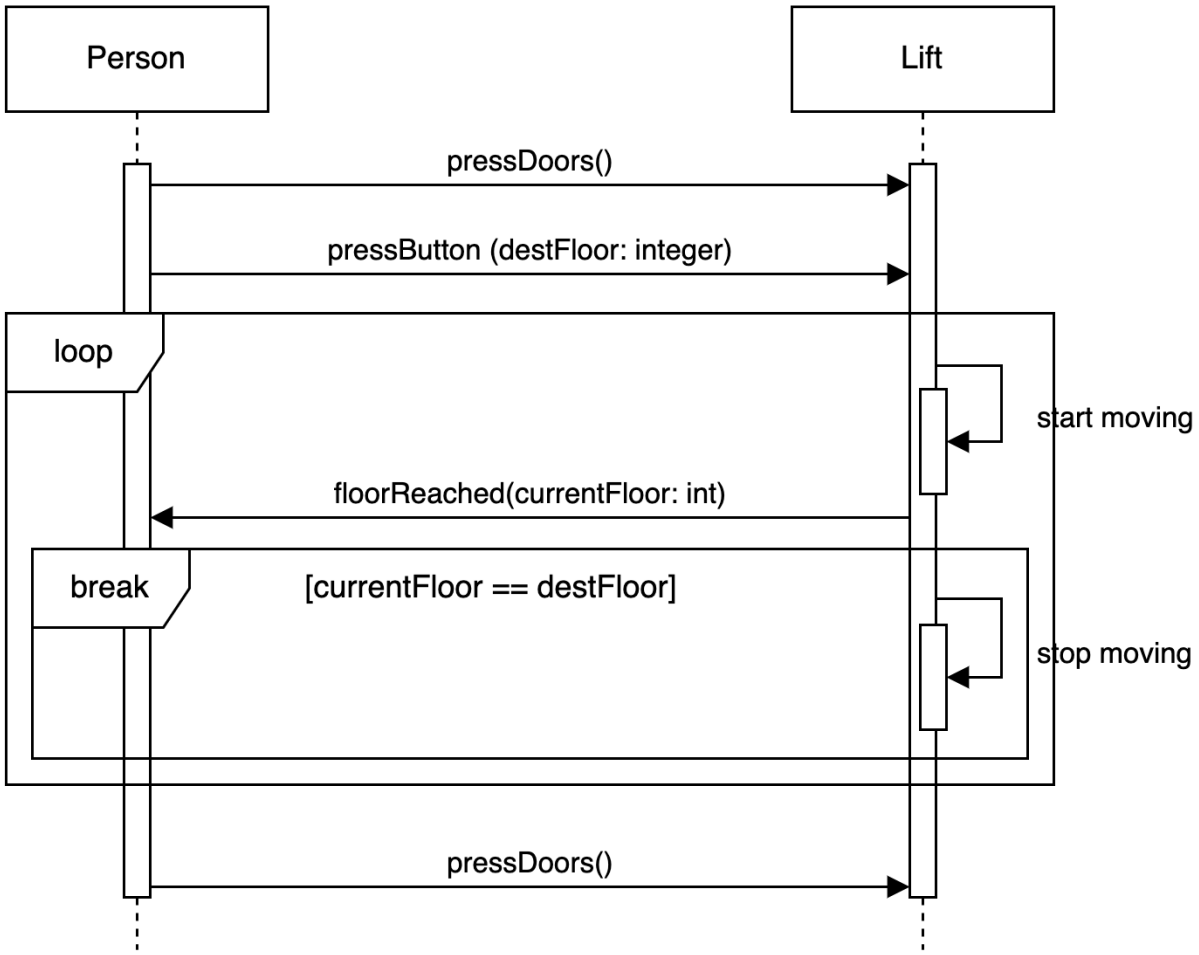
TASK 2 - A



Description:

We change the parameter of the pressButton method from integer to enum, where enum consists of values from 1 to 5. We also limit the number of executions in the cycle to a maximum of 5 repetitions.

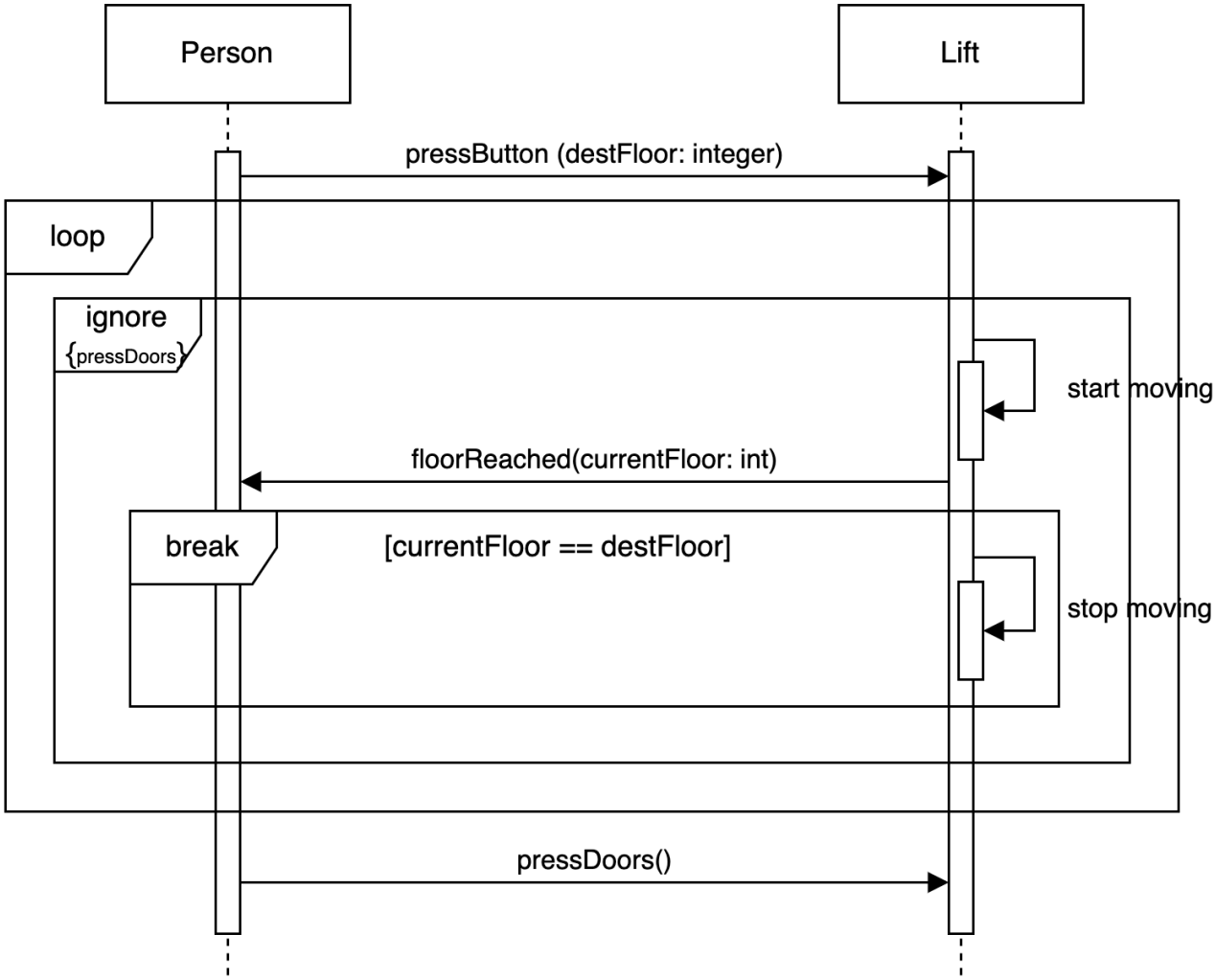
TASK 2 - B



Description:

Added pressDoors() before pressButton() is called.

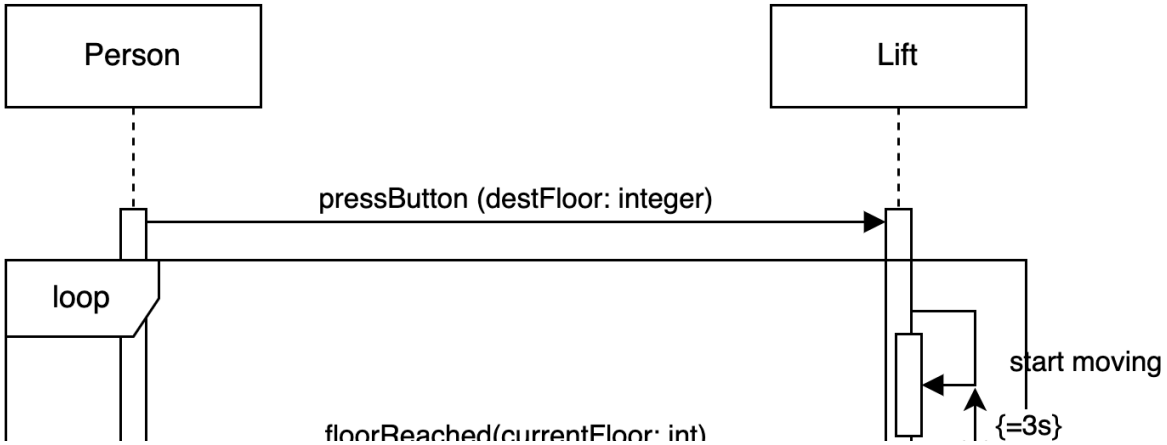
TASK 2 - C

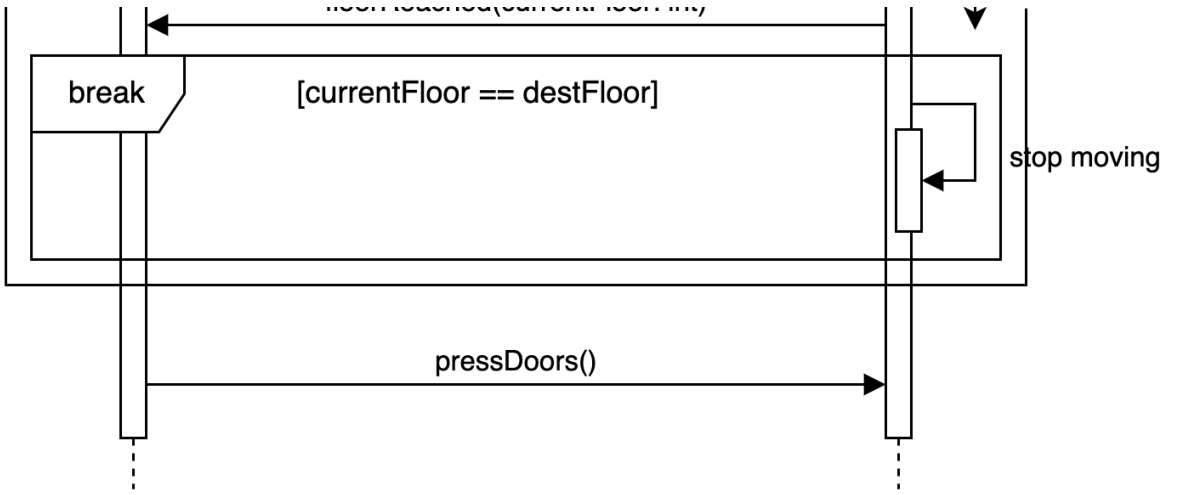


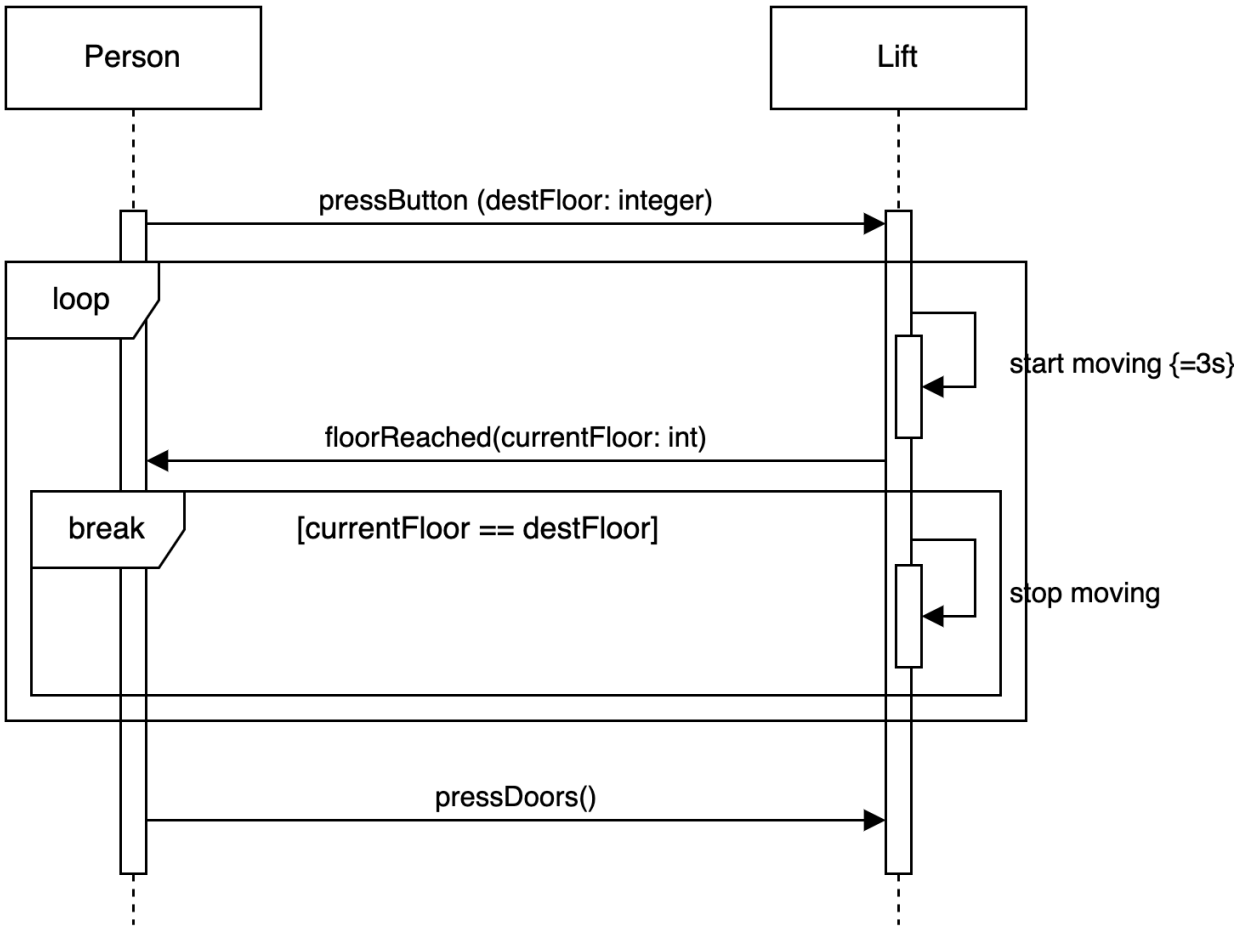
Description:

For each iteration of the loop we attach ignore to the call to pressDoors

TASK 2 - D



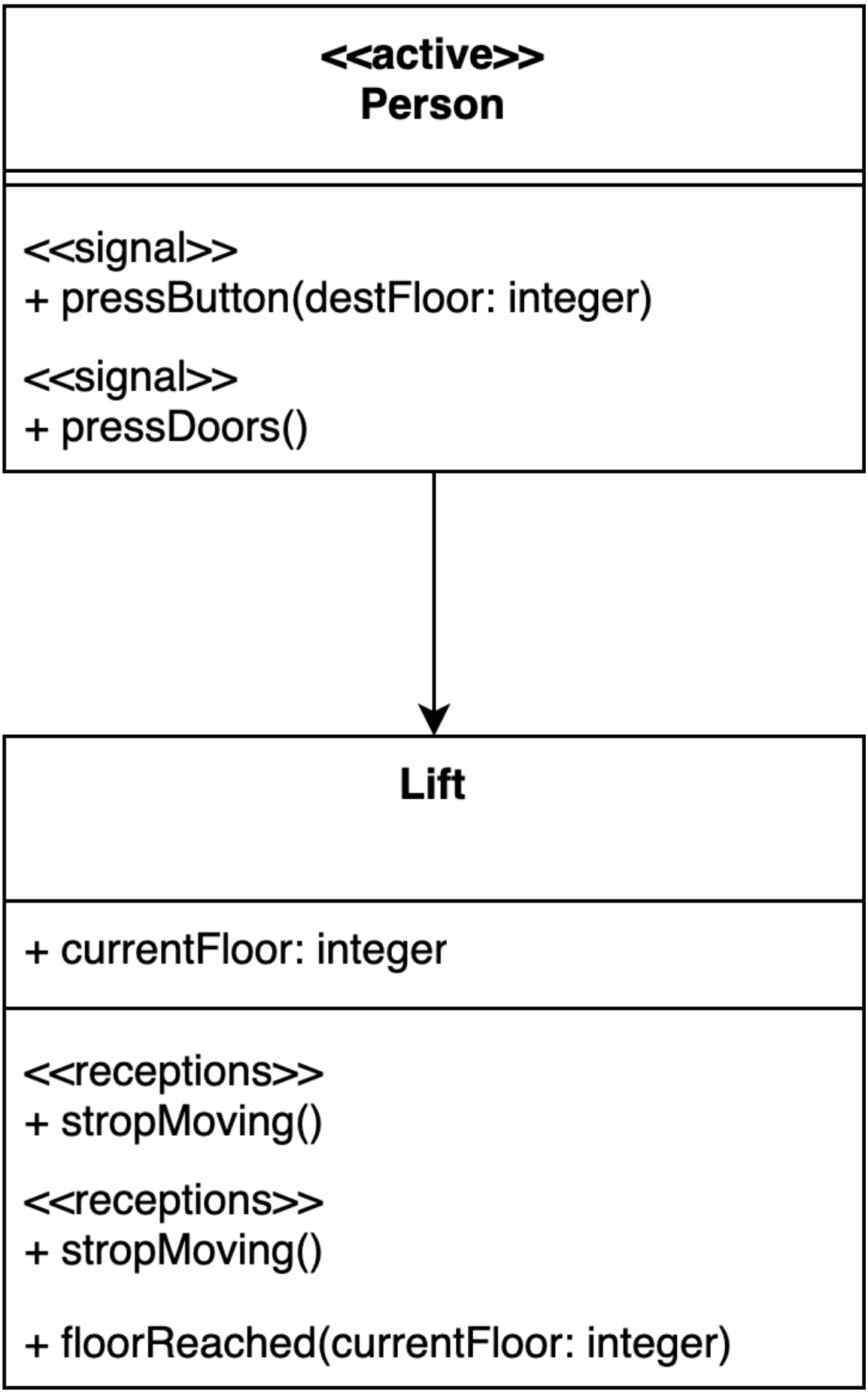




Description:

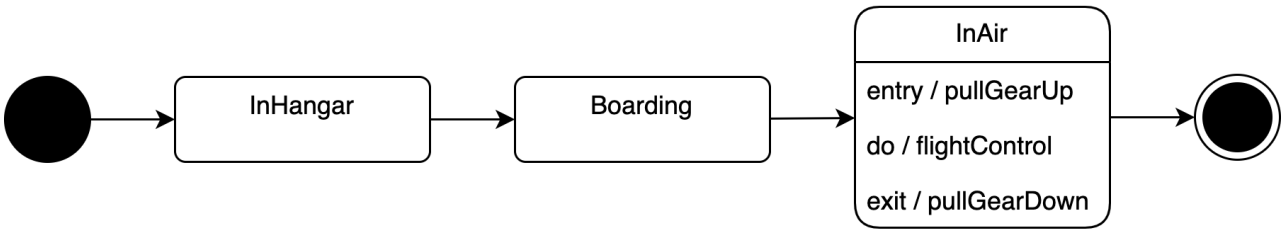
We impose a duration constraint on the execution of the iteration between the startMoving and floorReached operations

TASK 2 - E

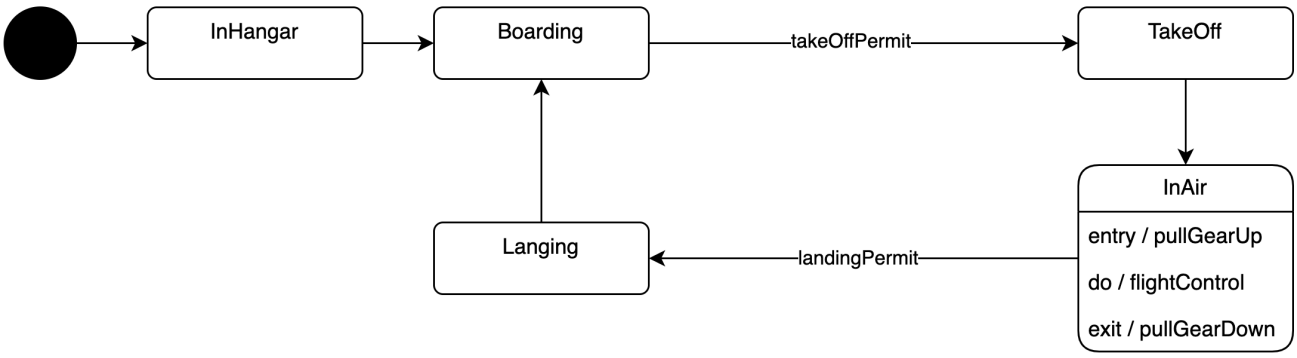

<<active>>
Person
<<signal>>
+ pressButton(destFloor: integer)
<<signal>>
+ pressDoors()
Lift
+ currentFloor: integer
<<receptions>>
+ stropMoving()
<<receptions>>
+ stropMoving()
+ floorReached(currentFloor: integer)

Task - 4

TASK 4 - BASE



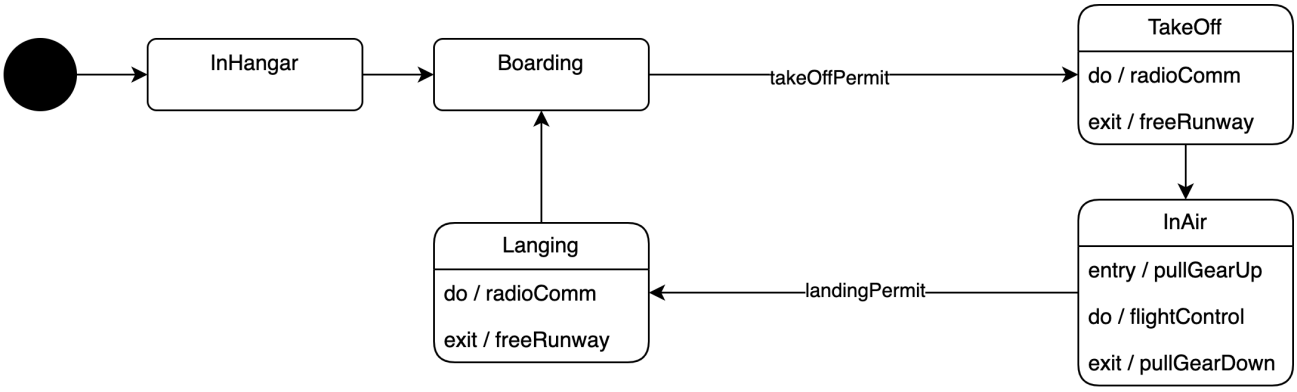
TASK 4 - A



Description:

Added new states TakeOff and Landing. The diagram is looped on the transition to Landing -> Boarding, because we have no other conditions under which the transitions by states will stop.

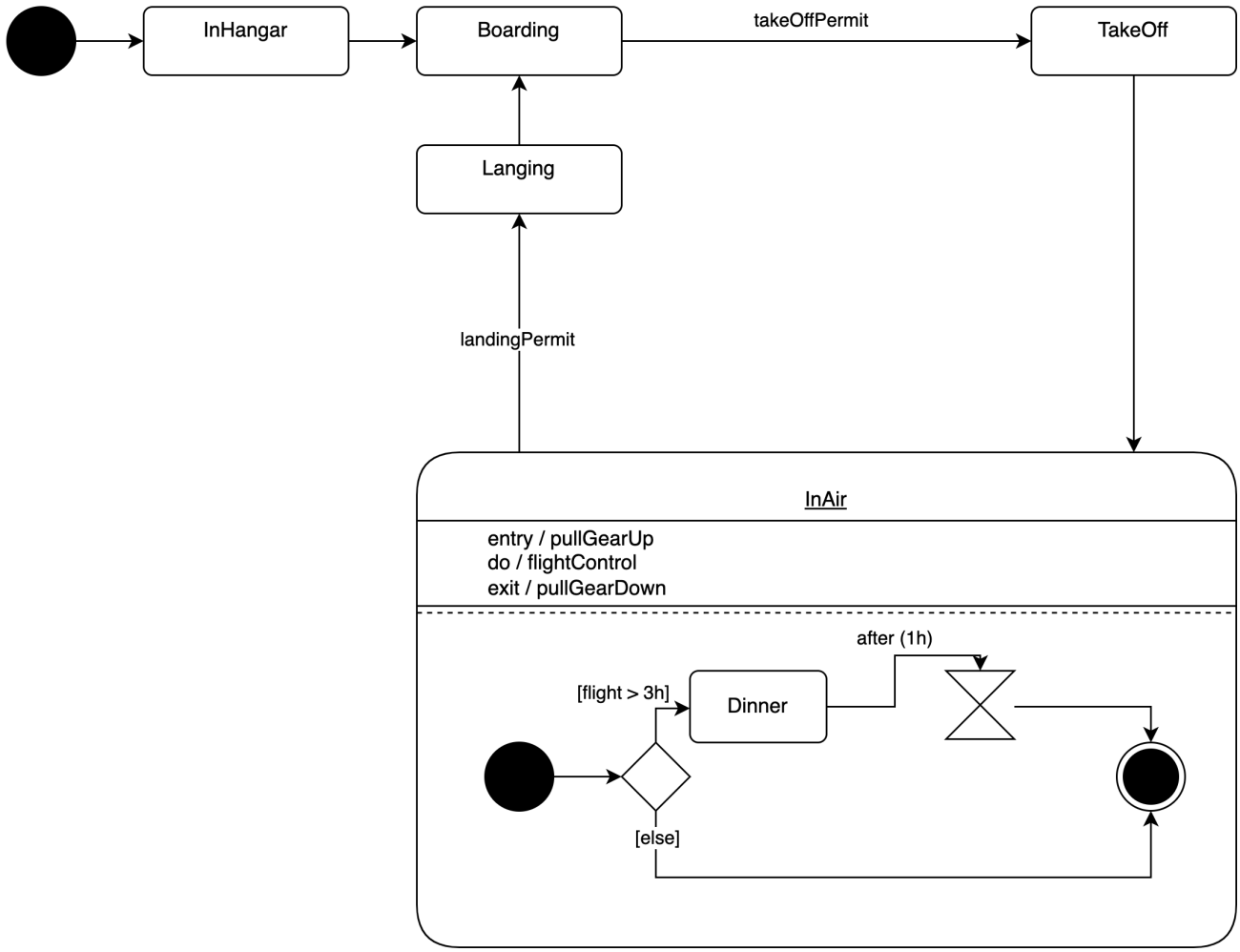
TASK 4 - B



Description:

Updated TakeOff and Landing states to "do / radioComm" and "exit / freeRunway"

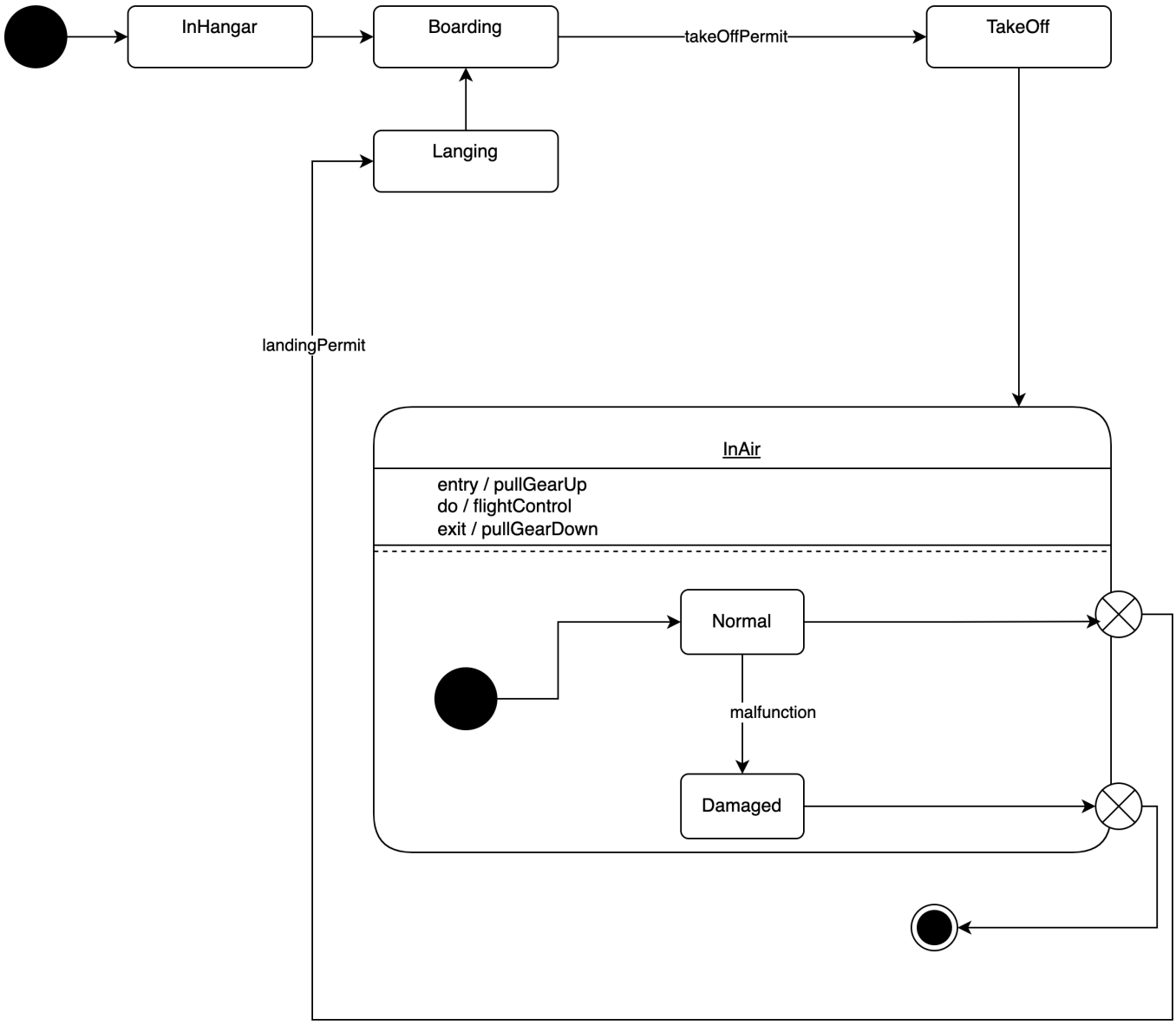
TASK 4 - C



Description:

We have transformed InAir into a complex state by adding a substate that is nested and orthogonal, which specifies a subchart for dinner service. The subchart includes a check to determine if the flight is expected to last more than 3 hours. If so, passengers are offered a meal at any point during the flight.

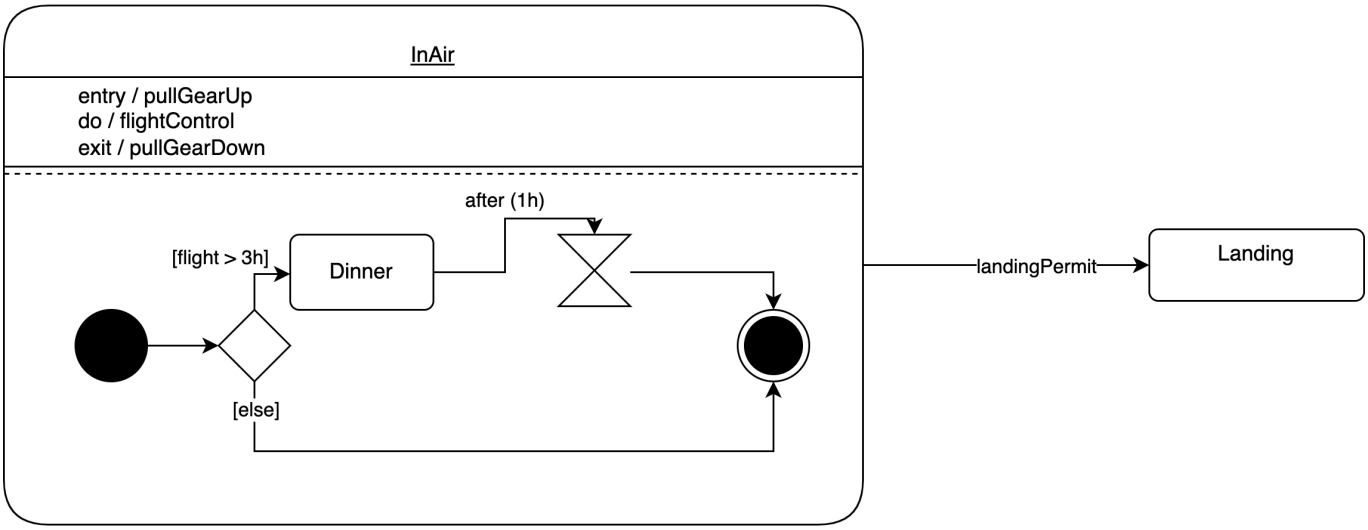
TASK 4 - D



Description:

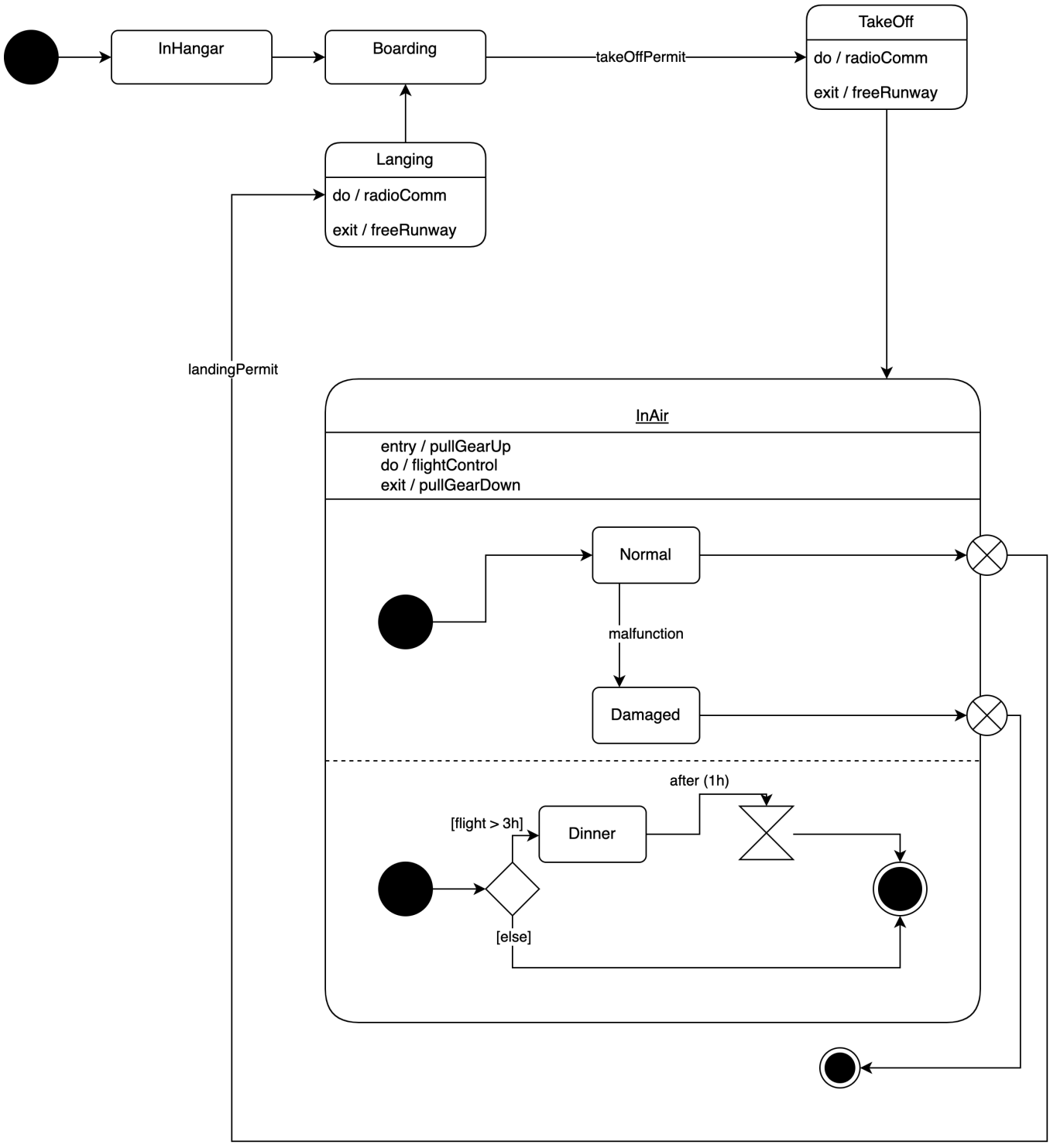
Similar to point C, a new orthogonal region is added, where the transition by substates from Normal to Damaged is indicated. Since the condition does not specify what happens to the aircraft after damage, we assume that the statechart stops at this point and everything goes to the final state

TASK 4 - E

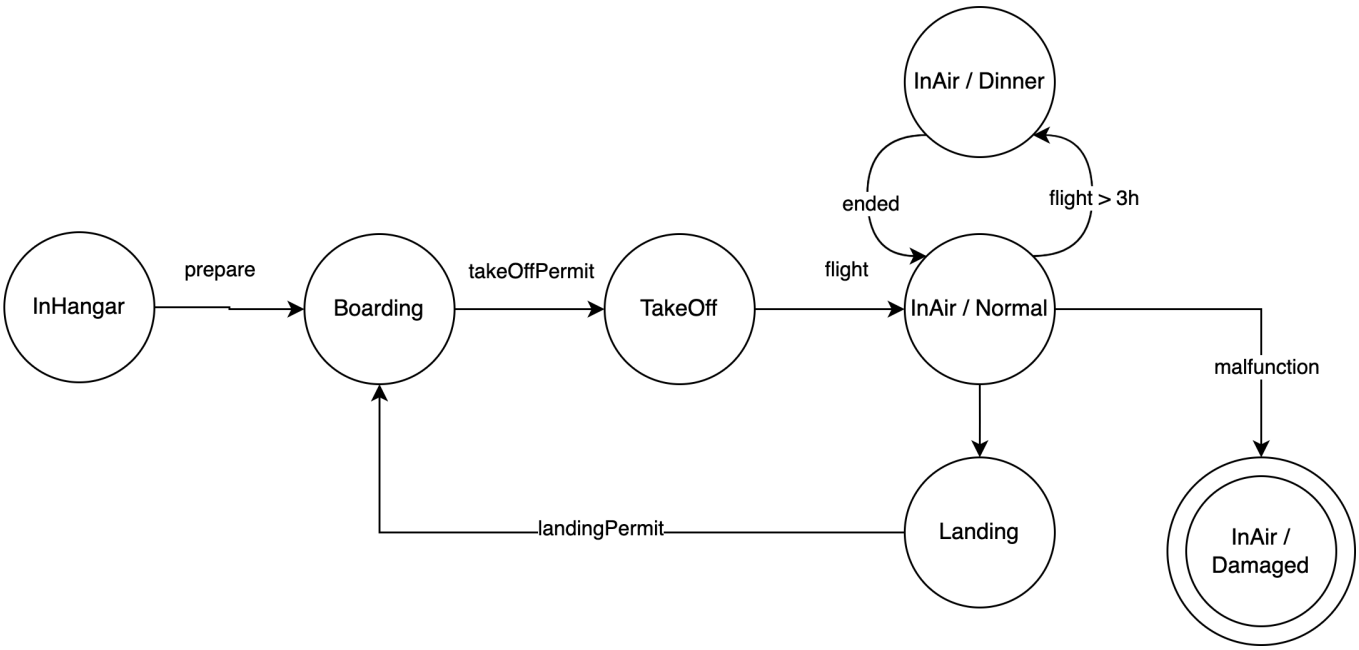


Dinner must be interrupted, since **dinner** is a substate of the complex state **InAir**, then when the **landingPermit** event arrives, the parent state InAir and all its substates will be interrupted.

TASK 4 - F

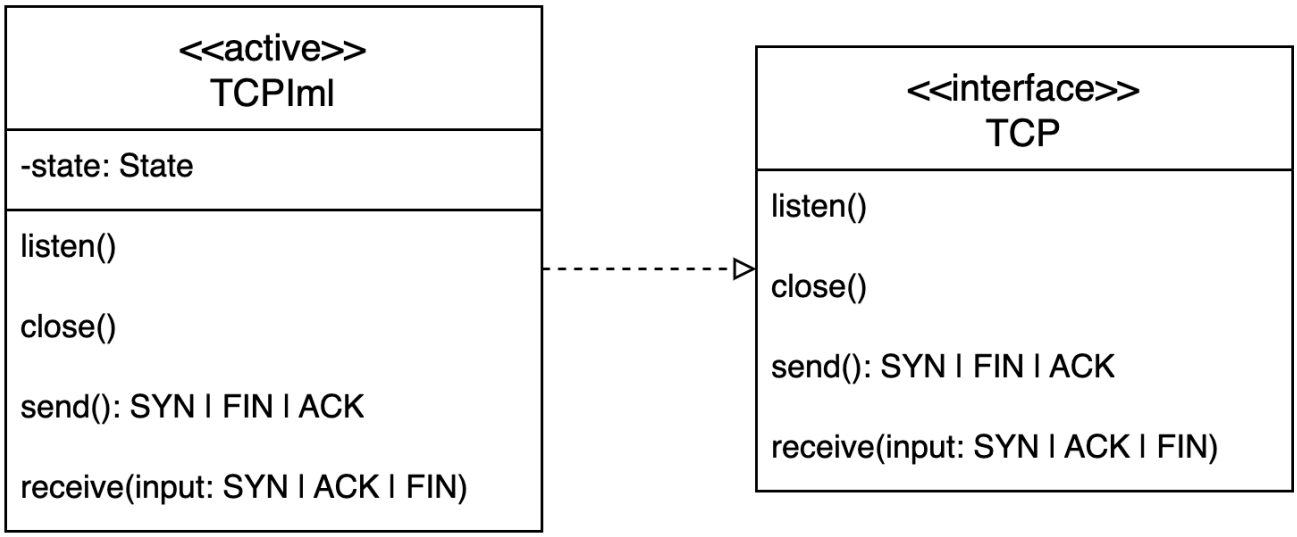

InHangar
Boarding
takeOffPermit
TakeOff
do / radioComm
exit / freeRunway
Langing
do / radioComm
exit / freeRunway
landingPermit
InAir
entry / pullGearUp
do / flightControl
exit / pullGearDown
Normal
malfunction
Damaged
after (1h)
[flight > 3h]
Dinner
[else]

TASK 4 - G

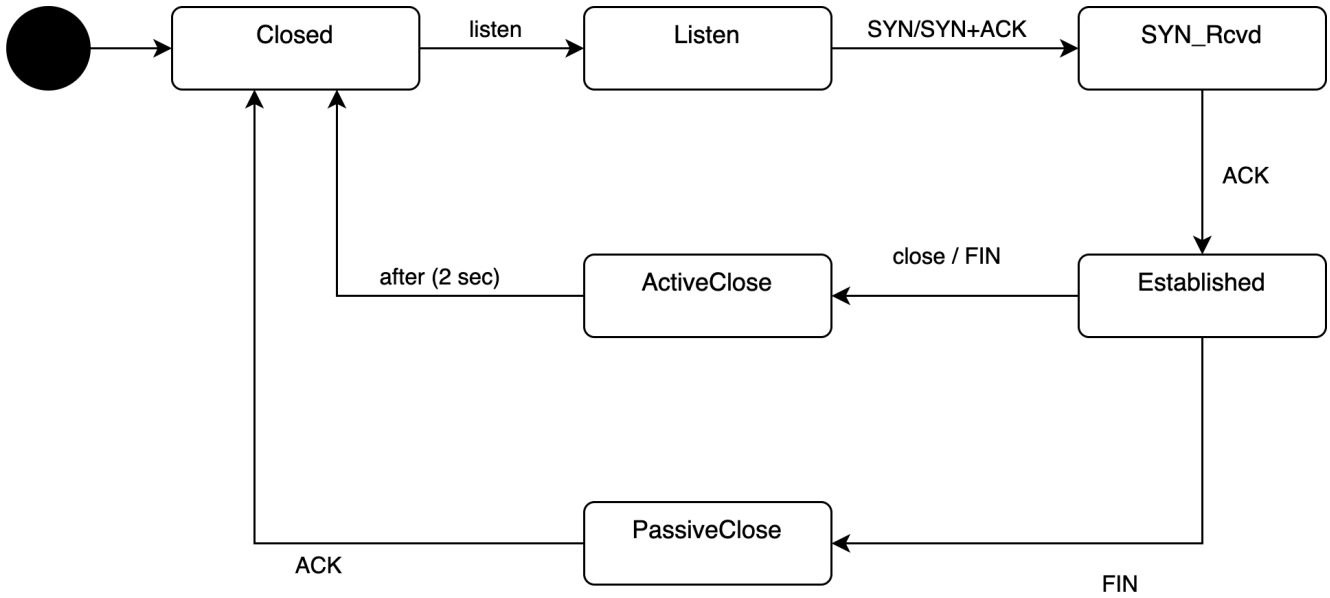


Task - 5

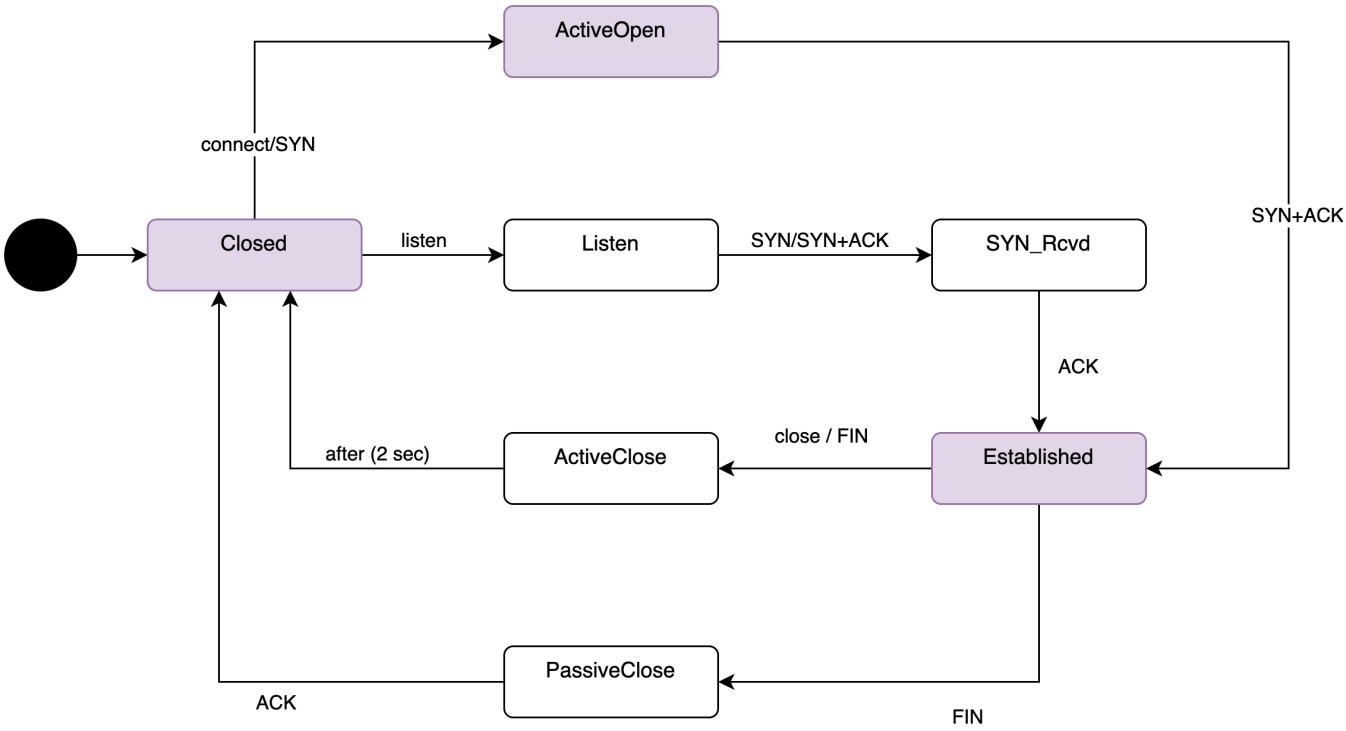
TASK 5 - BASE



TASK 5 - A



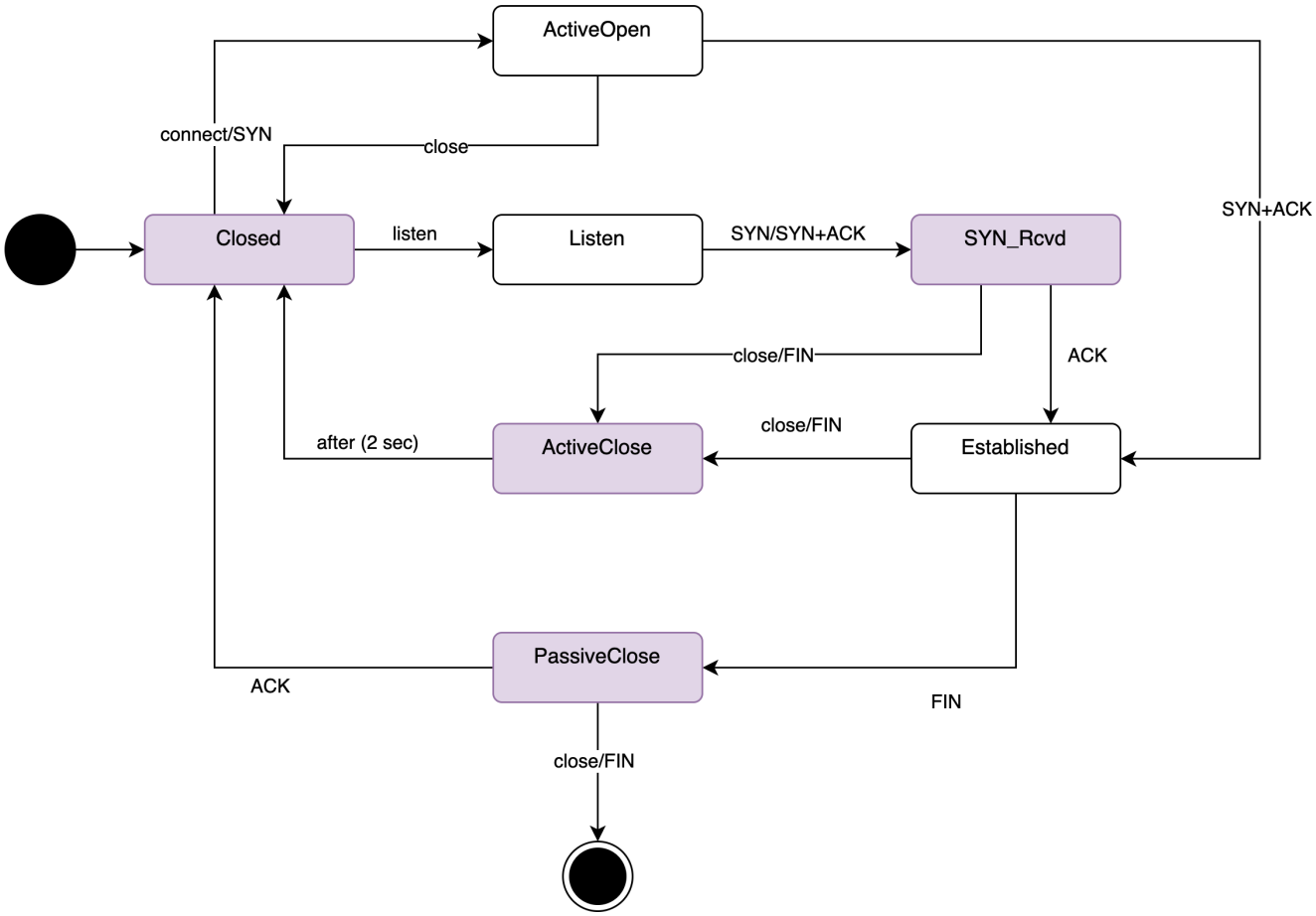
TASK 5 - B



Desciption:

Added a new transition from Closed to ActiveOpen and then to Established.

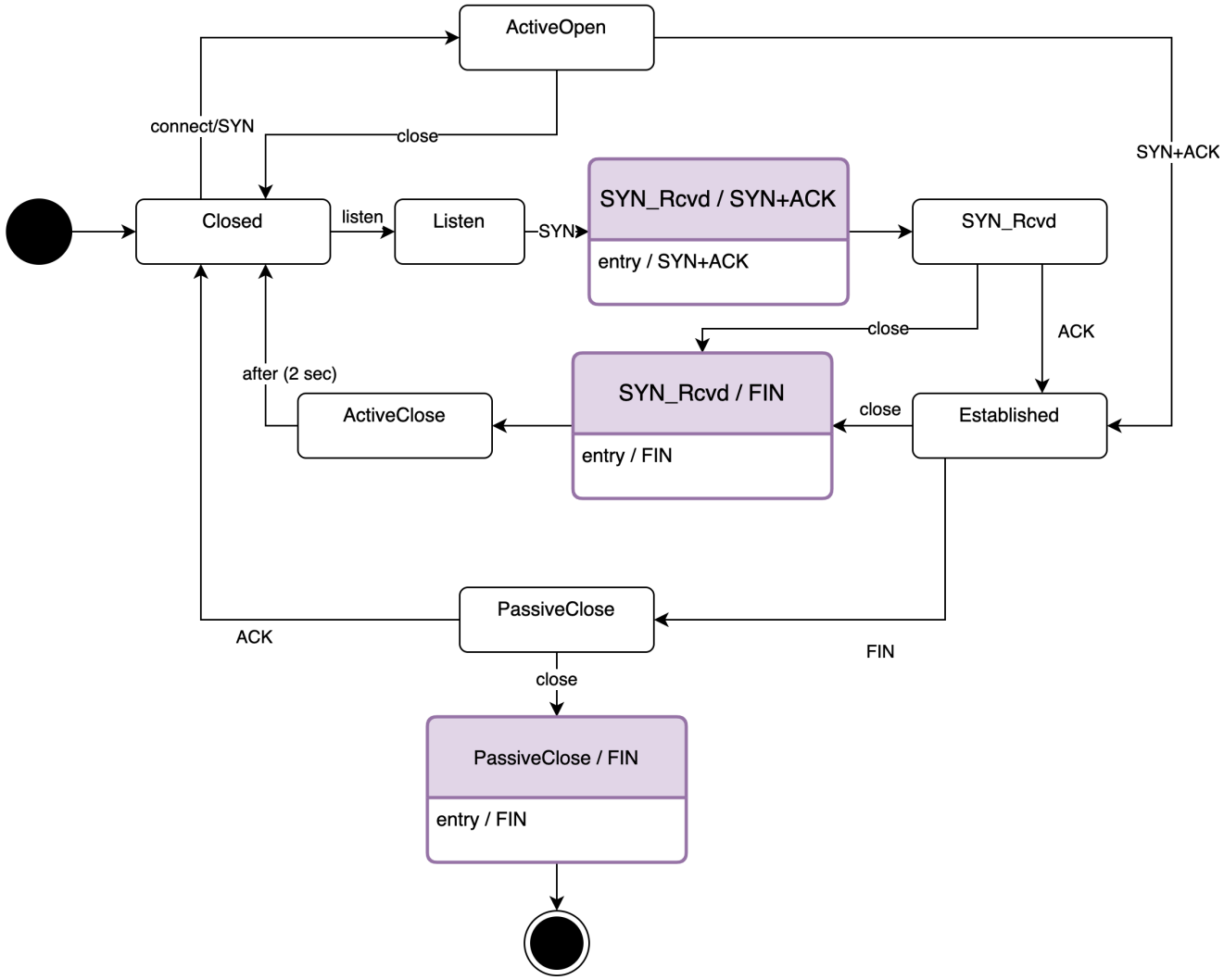
TASK 5 - C



Description:

Added an additional transition from the PassiveClose state to the FinalState by the close event, from the SYN_RCVD state by the close event added a new transition to the ActiveClose state, in addition, a new transition from the ActiveOpen state to the Closed state by the close event was added

TASK 5 - D



Description:

For transitions with effect, intermediate states were added, so the states SYN_RCVD / SYN + ACK were added on the transition from Listen to SYN_RCVD, SYN_RCVD / FIN on the transition from SYN_RCVD to ActiveClose, and PassieveClose / FIN on the transition from PassieveClose to FinalState