Air Lift

The described activities take place during passenger air lift from an origin town and a destination town somewhere in Portugal and aim to portray what happens in between. One assumes there are three relevant locations: the *departure airport*, where passengers arrive at random times to check in for the transfer flight; the *plane*, which takes them to their destination; and the *destination airport*, where they arrive after the travel.

Three types of entities will be considered: the *passengers*, who take the transfer flight between the two towns; the *hostess*, who controls the embarking procedure; and the *pilot*, who flies the plane.

A single plane and an indeterminate number of transfer flights are assumed: passengers are only transported from the origin town to the destination town; on the way back, the plane flies empty. A fixed number of *N* passengers will transfer. The number of passengers that take each flight is variable: it ranges between *MIN* and *MAX* for all the flights, but the last; for the last, all the remaining passengers are transported, whatever its number.

Activities are organized in the following way

- upon arrival at the departure airport, passengers queue in at transfer gate waiting to be called by the hostess;
- when advised by the pilot that the plane is ready to board, the hostess signals the passenger at the head of the queue, if there is one, to join her and present the flight documentation; after checking it, she requests him/her to board the plane
- if the queue is empty and the number of passengers that have already boarded is between *MIN* and *MAX*, or there are no more passengers to transfer, the hostess advises the pilot that the boarding is complete and that the the flight may start;
- once inside the plane, the passengers take a seat and wait for the flight to take place; upon arrival, they exit the plane and leave the airport;
- if there are still passengers to transfer, the pilot parks the plane at the departure gate and advises the hostess that the boarding may start; upon being informed that the boarding is complete, she flies to the destination town;
- after landing and parking the plane at the arrival gate, the pilot announces the passengers that they may leave; when the last passenger has exited, she flies back to the origin town.

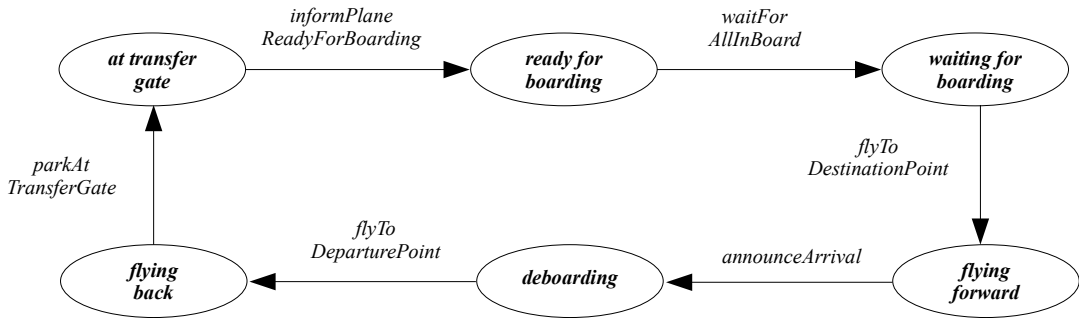
In the end of the day, a full report of the activities is issued.

Assume that there are twenty one passengers participating in the air lift, that the plane capacity is ten and that a transfer flight takes place if there is a minimum of five passengers. Write a simulation of the life cycle of the passengers, the hostess and the pilot using one of the models for *thread* communication and synchronization which have been studied: monitors, or semaphores and shared memory.

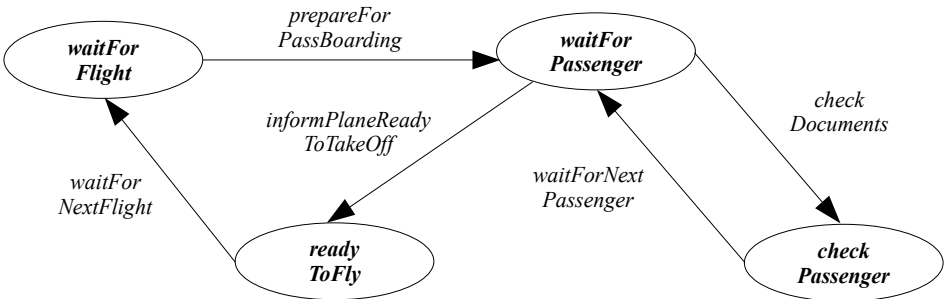
One aims for a distributed solution with multiple information sharing regions, written in Java, run in Linux and which terminates. A *logging* file, which describes the evolution of the internal state of the problem in a clear and precise way, must be included.

Suggestion to solution

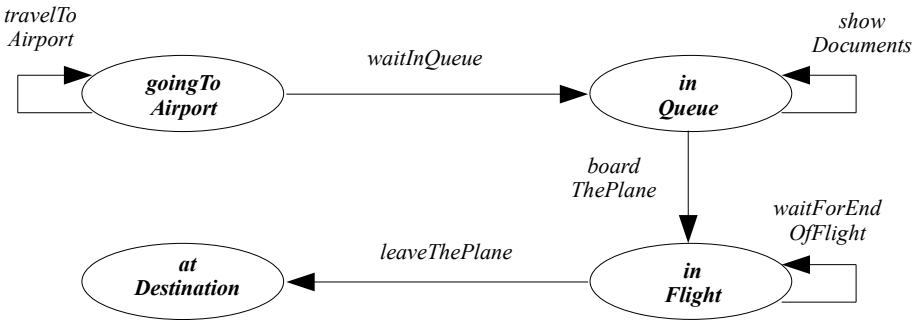
Life cycle of the pilot



Life cycle of the hostess



Life cycle of the passenger



Characterization of the interaction

Pilot

AT_TRANSFER_GATE – transition state (initial / final state)
READY_FOR_BOARDING – transition state
WAIT_FOR_BOARDING – blocking state
The pilot is waken up by the operation *informPlaneReadyToTakeOff* of the hostess when boarding is complete.
FLYING_FORWARD – independent state with blocking
The pilot should be made to sleep for a random time interval in the simulation.
DEBOARDING – blocking state
The pilot is waken up by the operation *leaveThePlane* of the last passenger to leave the plane.
FLYING_BACK – independent state with blocking
The pilot should be made to sleep for a random time interval in the simulation.

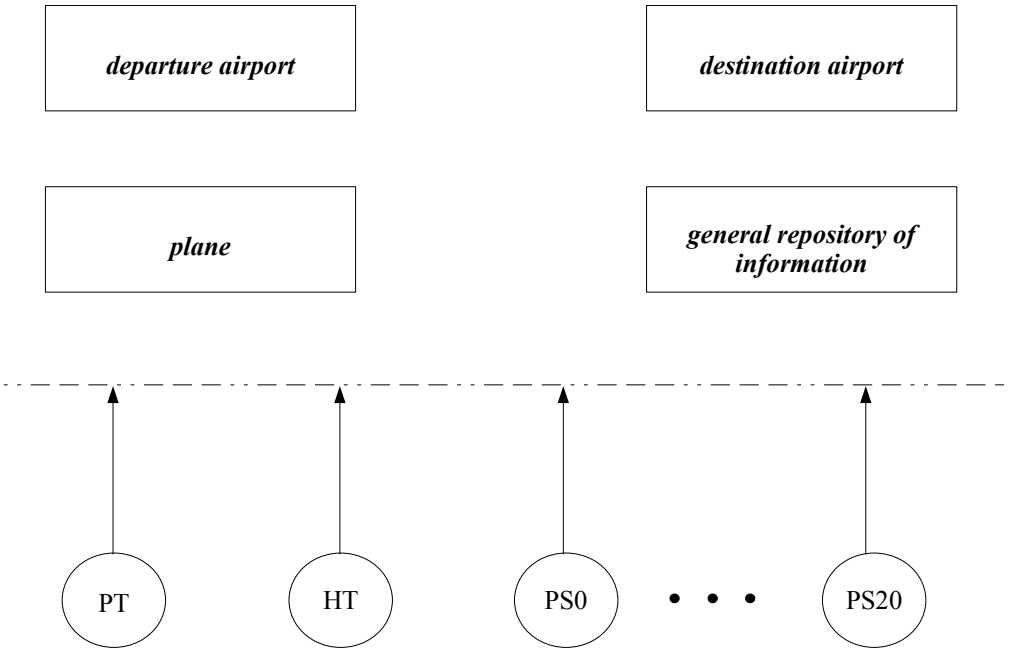
Hostess

WAIT_FOR_NEXT_FLIGHT – blocking state (initial / final state)
The hostess is waken up by the operation *informPlaneReadyForBoarding* of the pilot after parking the plane at the departure gate.
WAIT_FOR_PASSENGER – blocking state
The hostess is waken up by the operation *waitInQueue* of the passenger while he / she waits to have his / her documents checked.
CHECK_PASSENGER – blocking state
The hostess is waken up by the operation *showDocuments* of the passenger when he / she hands his / her plane ticket to be checked.
READY_TO_FLY – transition state

Passenger

GOING_TO_AIRPORT – independent state with blocking (initial state)
The passenger should be made to sleep for a random time interval in the simulation.
IN_QUEUE – double blocking state
The passenger is waken up first by the operation *checkDocuments* of the hostess requesting him/ her to present the plane ticket and is waken up next by the operation *waitForNextPassenger* of the hostess after the checking is being made.
IN_FLIGHT – blocking state
The passenger is waken up by the operation *announceArrival* of the pilot after parking the plane at the arrival gate.
AT_DESTINATION – transition state (final state)

Information sharing regions



Guidelines for solution implementation

1. Characterize interaction at the state level.

2. Specify the life cycle and internal properties of each of the *intervening entities*.

3. Specify for each *information sharing region* the internal data structure, the operations which will be invoked, identifying their signature, functionality and who is the calling entity, and the synchronization points.

4. Sketch the *interaction diagram* which describes in a compact, but precise, way the dynamics of your solution. Go back to steps 1 and 2 until you are satisfied the description is correct.

5. Proceed to its coding in Java as specific reference data types.

6. Write the application main program which should instantiate the different *information sharing regions* and the different *intervening entities*, then start the different entities and finally wait for their termination.

7. Validate your solution by taking several runs and checking for each, through the detailed inspection of the logging file, that the output data is indeed correct.