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 the client-server model with server replication, where the passengers, the hostess and the pilot are the *clients* and the access to the information sharing regions are the services provided to them by the *servers*.

The operations that were assigned to activities previously carried out in the information sharing regions (for the already implemented concurrent version), must now be assigned to independent requests performed on the servers, seen as remote objects, through the remote method of invocation.

One aims for a solution to be written in Java, to be run in Linux under Java RMI, either in a concentrated manner (on a single platform), or in a distributed fashion (up to 9 different platforms) and to terminate (it must contemplate service shutdown). A *logging* file, that describes the evolution of the internal state of the problem in a clear and precise way, must be included.

Guidelines for solution implementation

1. Specify for each remote object the interface of the methods to be invoked on it.
2. Specify the general organization of the servers architecture.
3. Specify the general organization of the clients architecture.
4. Sketch the interaction diagram which describes in a compact, but precise, way the dynamics of your solution. Go back to steps 1, 2 and 3 until you are satisfied the description is correct.
5. Proceed to its coding in Java as specific reference data types.
6. Specify the mapping of the servers and the clients onto multiples nodes of the parallel machine and write the shell scripts which enable the deployment and the execution of the different modules the application is composed of.
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.