Airport rhapsody

The described activities take place at an airport, somewhere in Portugal, and aim to portray what happens when passengers arrive from a flight. There are eight main locations: the arrival lounge, the baggage collection point, the temporary storage area (for holding the luggage of passengers in transit), the baggage reclaim office, the terminal transfer quays, the arrival terminal exit and the departure terminal entrance.

There are three types of entities: the *passengers*, who terminate their voyage at the airport or are in transit, the *porter*, who unloads the the bags from a plane, when it lands, and carries them to the baggage collection point or to the temporary storage area, and the *bus driver*, who moves the passengers in transit between the arrival and the departure terminals.

K plane landings are assumed, each involving the arrival of N passengers. Each passenger carries 0 to M pieces of luggage in the plane hold. The bus, which moves the passengers between terminals, has a capacity of T seating places.

Activities are organized, for each plane landing, in the following way

- the passengers walk from the arrival lounge to the baggage collection point, if their journey ends at this airport and have bags to collect; those without bags go directly to the arrival terminal exit and leave the airport; the remaining passengers, who are in transit, walk to the terminal transfer quay;
- after all the passengers have left the plane, the porter picks up the pieces of luggage, one by one, from the plane hold and carries them either to the baggage collection point, or to the temporary storage area, as they belong to local or in transit passengers, respectively;
- in the baggage collection point, the passengers wait for the arrival of their bags; upon taking possession of them, they walk to the arrival terminal exit and leave the airport; those with missed bags go first to the baggage reclaim office to post their complaint, before walking to the arrival terminal exit and leave the airport;
- on the terminal transfer quay, the passengers wait for the bus arrival, which will take them to the departure terminal for the next leg of the journey;
- the bus leaves the terminal transfer quay according to a predefined schedule, executing a circular path which has as another stop the terminal transfer quay of the departure terminal; however, if it happens that all seats are occupied prior to the predefined time to leave, the driver may depart sooner.

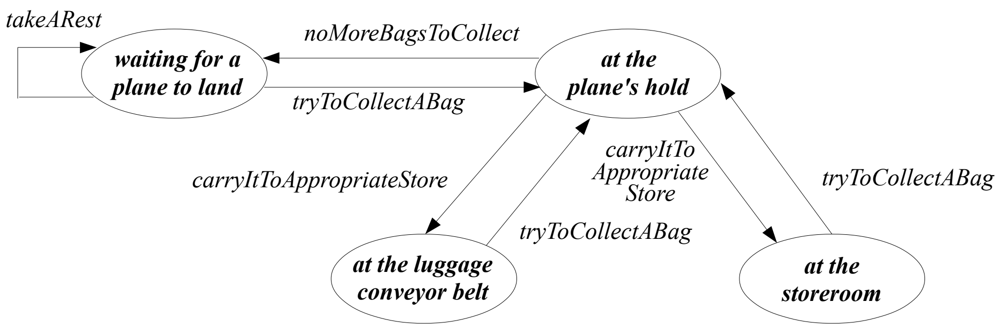
In the end of the day, a full report of the activities is issued.

Assume that there are five plane landings, each involving the arrival of six passengers, carrying a maximum of two pieces of luggage in the plane hold and that the transfer bus has a capacity of three seating places. Write a simulation of the life cycle of the passengers, the porter and the bus driver using one of the models for *thread* communication and synchronization which have been studied: monitors or semaphores and shared memory.

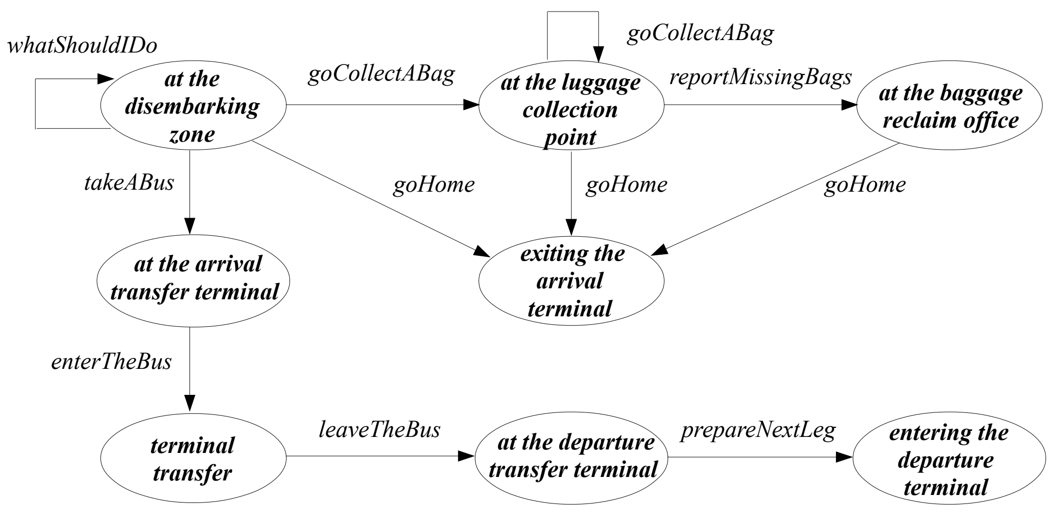
One aims for a distributed solution with multiple information sharing regions, written in Java, run in Linux and which terminates. A *logging* file, which describes the evolution of the internal state of the problem in a clear and precise way, must be included.

Suggestion to solution

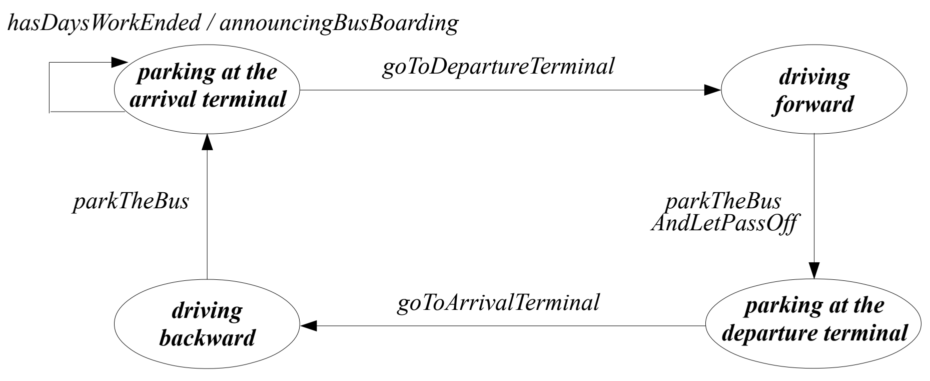
Life cycle of the porter



Life cycle of the passenger


whatShouldIDo
at the disembarking zone
goCollectABag
goCollectABag
at the luggage collection point
reportMissingBags
at the baggage reclaim office
takeABus
goHome
goHome
goHome
at the arrival transfer terminal
exiting the arrival terminal
enterTheBus
terminal transfer
leaveTheBus
at the departure transfer terminal
prepareNextLeg
entering the departure terminal

Life cycle of the bus driver



Characterization of the interaction

<u>*Porter*</u>

WAITING_FOR_A_PLANE_TO_LAND – blocking state (initial / final state)
The porter is waken up by the operation *whatShouldIDo* of the last of the passengers to reach the arrival lounge
AT_THE_PLANES_HOLD – transition state
AT_THE_LUGGAGE_BELT_CONVEYOR – transition state
AT_THE_STOREROOM – transition state

<u>*Passenger*</u>

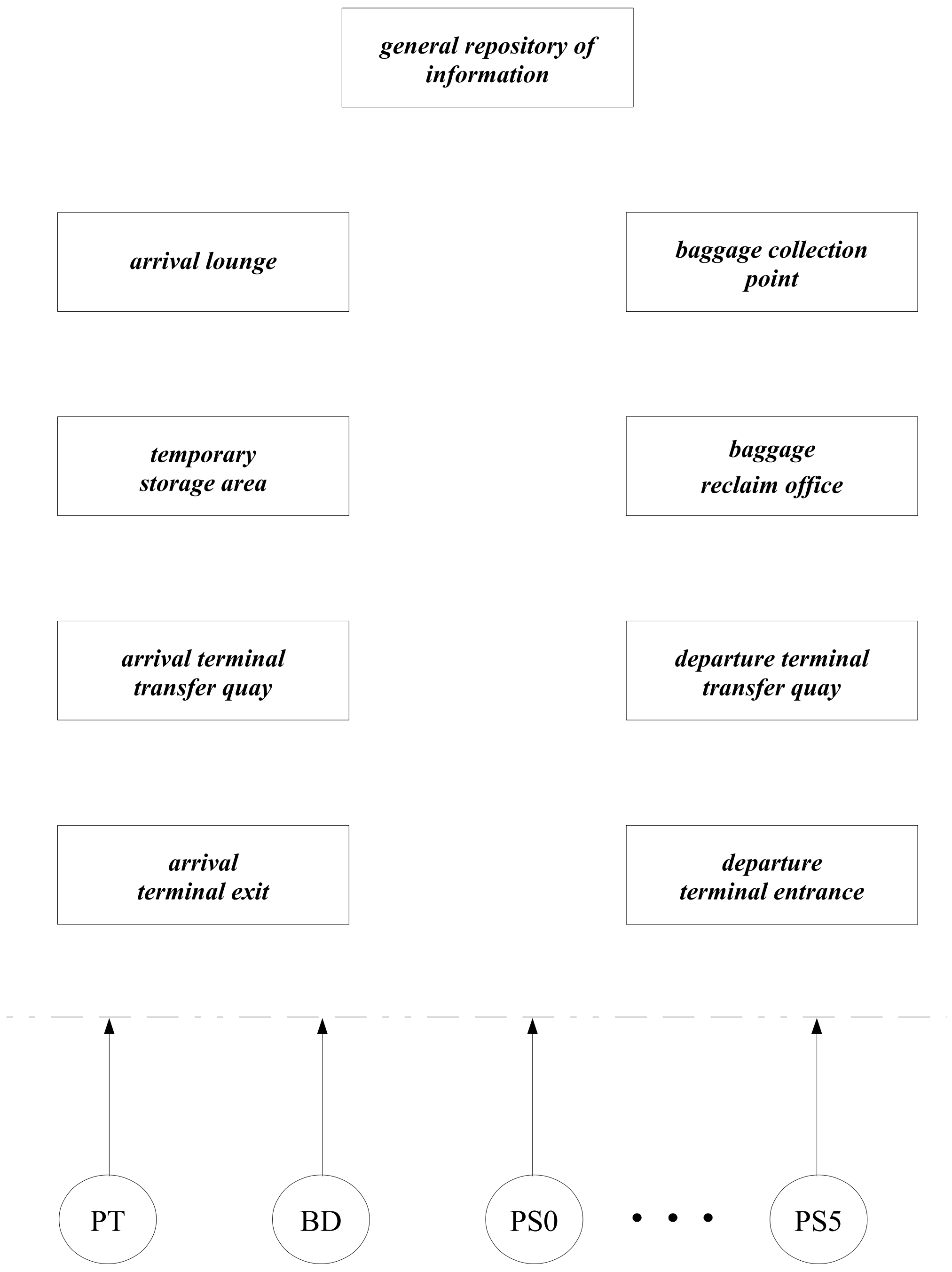
AT_THE_DISEMBARKING_ZONE – transition state (initial state)
AT_THE_LUGGAGE_COLLECTION_POINT – blocking state with eventual transition
the passenger is waken up by the operations *carryItToAppropriateStore* and *tryToCollectABag* of the porter when he places on the conveyor belt a bag she owns, the former, or when he signals that there are no more pieces of luggage in the plane hold, the latter, and makes a transition when either she has in her possession all the bags she owns, or was signaled that there are no more bags in the plane hold
AT_THE_BAGGAGE_RECLAIM_OFFICE – transition state
EXITING_THE_ARRIVAL_TERMINAL – blocking state with eventual transition (final state)
the passenger is waken up by the operations *goHome* or *prepareNextLeg* of the last passenger of each flight to exit the arrival terminal or to enter the departure terminal
AT_THE_ARRIVAL_TRANSFER_TERMINAL – blocking state
before blocking, the passenger wakes up the bus driver, if her place in the waiting queue equals the bus capacity, and is waken up by the operation *announcingBusBoarding* of the driver to mimic her entry in the bus
TERMINAL_TRANSFER – blocking state
the passenger is waken up by the operation *parkTheBusAndLetPassOff* of the driver
AT_THE_DEPARTURE_TRANSFER_TERMINAL – transition state
ENTERING_THE_DEPARTURE_TERMINAL – blocking state with eventual transition (final state)
the passenger is waken up by the operations *goHome* or *prepareNextLeg* of the last passenger of each flight to exit the arrival terminal or to enter the departure terminal

<u>*Bus driver*</u>

PARKING_AT_THE_ARRIVAL_TERMINAL – double blocking state (initial / final state)
the driver is waken up the first time by the operation *takeABus* of the passenger who arrives at the transfer terminal and finds out her place in the waiting queue equals the bus capacity, or when the departure time has been reached (transition will only occurs if there is at least one passenger forming the queue); the driver is waken up the second time by the operation *enter TheBus* of the last passenger to enter the bus
DRIVING_FORWARD – transition state
PARKING_AT_THE_DEPARTURE_TERMINAL – blocking state
the driver is waken up by the operation *leaveTheBus* of the last passenger to exit the bus
DRIVING_BACKWARD – transition state

Information sharing regions

entrada


general repository of information
arrival lounge
baggage collection point
temporary storage area
baggage reclaim office
arrival terminal transfer quay
departure terminal transfer quay
arrival terminal exit
departure terminal entrance
PT
BD
PS0
PS5

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.