

## 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 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.

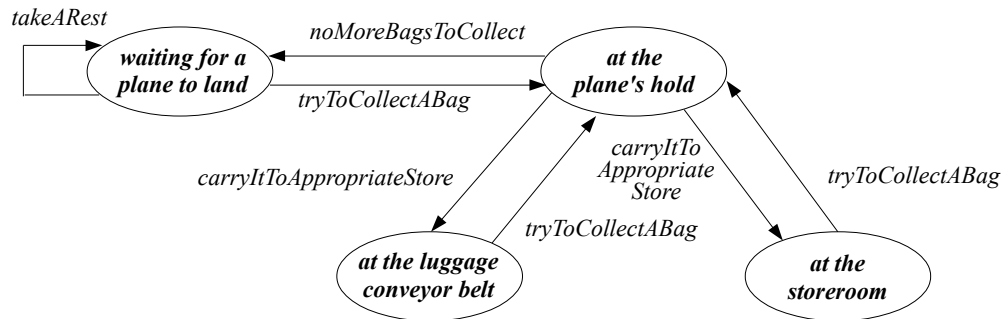
In the end of the day, a full report of the activities is issued.

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.

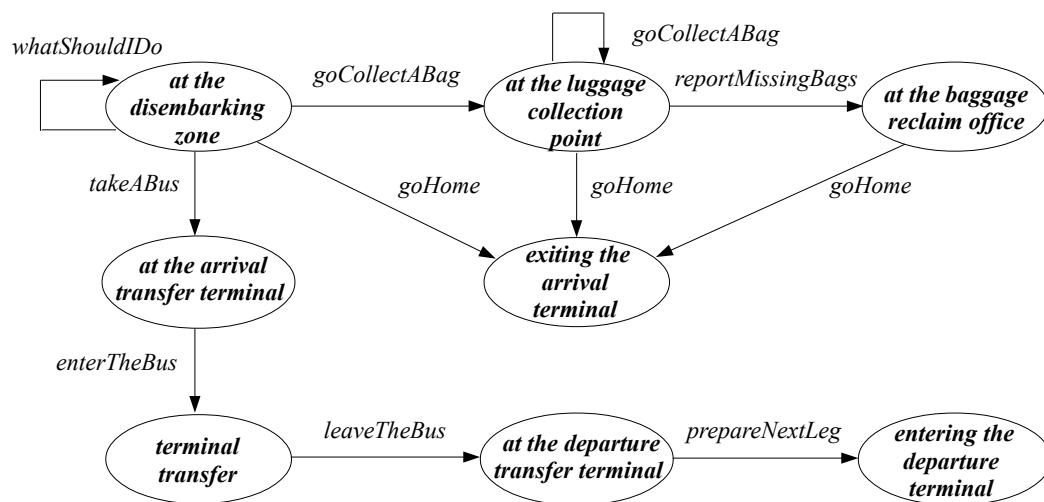
One aims for a distributed solution with multiple information sharing regions, written in Java, run in Linux and which terminates. A logging file, which describes the evolution of the internal state of the problem in a clear and precise way, must be included.

### Suggestion to solution

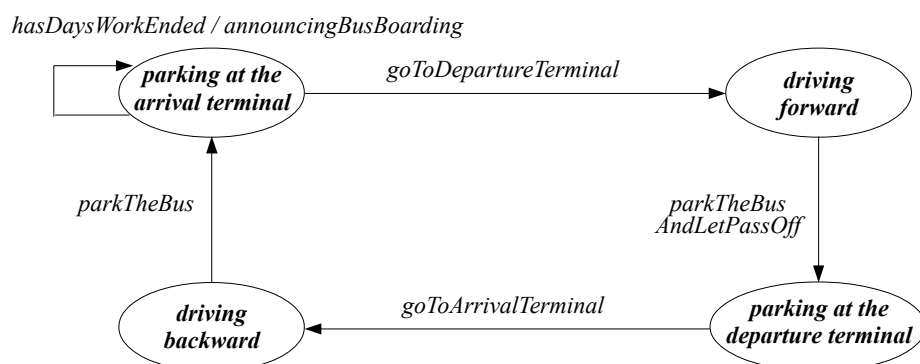
#### Life cycle of the porter



#### Life cycle of the passenger



#### Life cycle of the bus driver



## Characterization of the interaction

### Porter

*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

### Passenger

*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

### Bus driver

*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 *enterTheBus* of the last passenger to enter the bus

*DRIVING\_FORWARD* – transition state

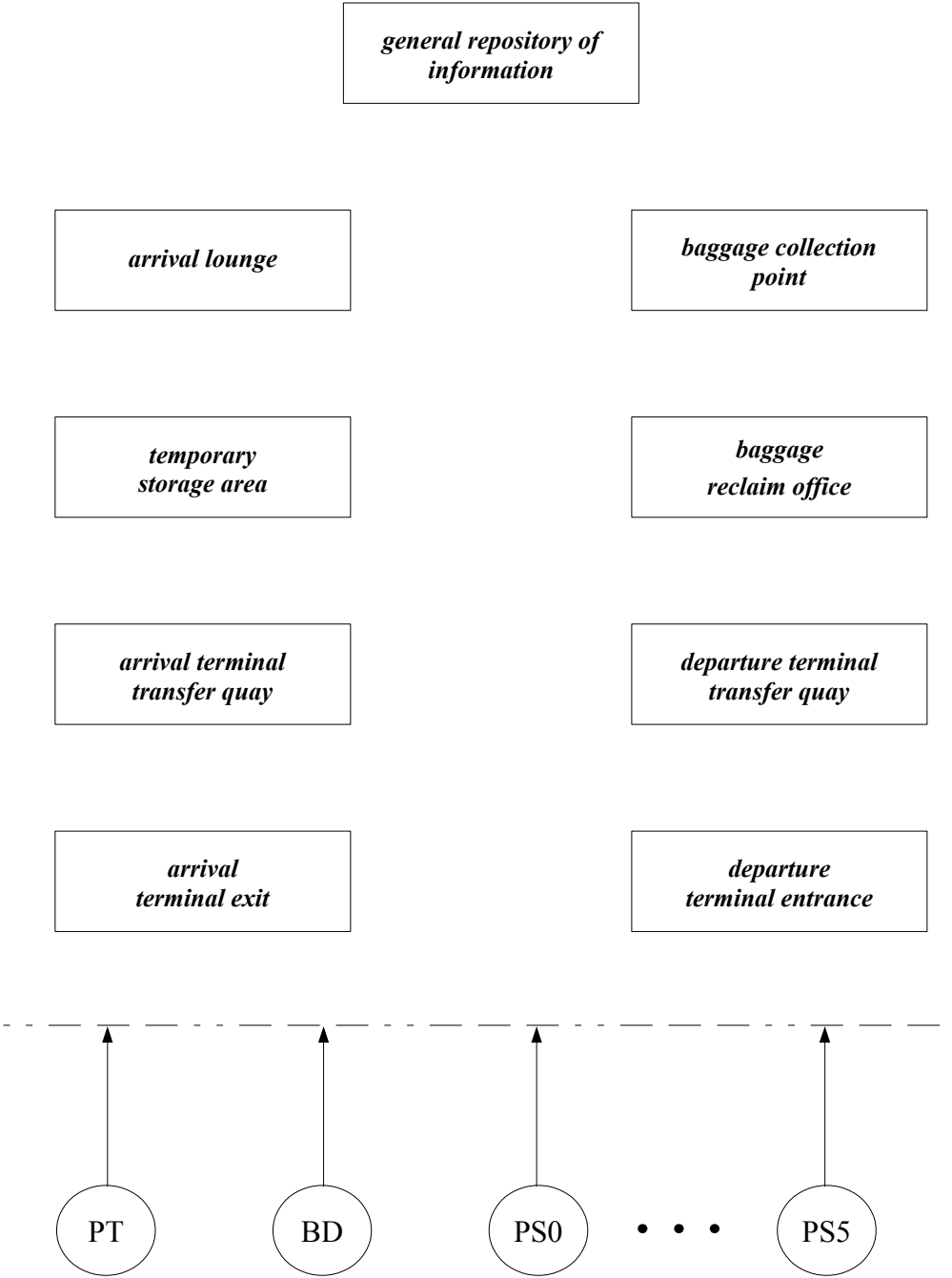
*PARKING\_AT\_THE\_DEPARTURE\_TERMINAL* – blocking state

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*



***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.