The restaurant

Events portray the activities that take place when a group of students, enrolled in *Computação Distribuída*, go to a famous restaurant downtown for a gourmet dinner to celebrate the beginning of the second semester. There are three main locations within the restaurant that should be accounted for: the *table* where the students sit to have their meal, the *kitchen* where the chef prepares it according to the orders placed by the students, and the *bar* where the waiter stands waiting for service requests. There are, furthermore, three kinds of interacting entities: N *students* , one *waiter* and one *chef*.

The activities are organized as described below

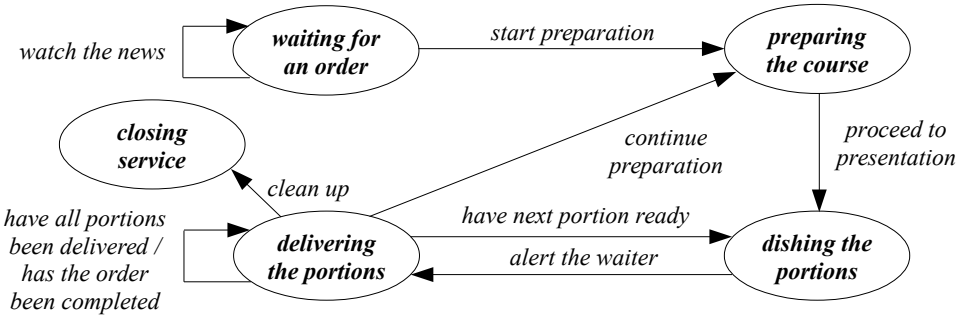
- the students arrive one by one at random times and sit at the table, chatting with one another while waiting for the group to be complete;
- once a students sits, the waiter brings her/him a copy of the menu so that she/he can select among the offered dishes the ones of her/his preference;
- the first student to arrive gathers the individual plate choices of her/his companions and prepares the order for the whole group;
- once the order has been completed, she/he calls the waiter and informs him about its content;
- the waiter, then, goes to the kitchen and places the order to the chef;
- the order consists of M courses per participant in the dinner;
- the waiter serves them in succession, only passing to the next course when signaled by the last student to finish eating that everybody is ready;
- in the end, the student that was the last to arrive signals the waiter to bring her/him the bill after both her/his companions and her/himself have finished the dessert and pays it in full as a form of penalty for being late;
- all students leave together the restaurant and go home to study because an assignment deadline is soon due.

Assume there are seven students and that the order consists of three courses per participant in the dinner: a starter, a main course and a dessert. Write a simulation of the life cycle of the chef, the waiter and the students using one of the models for *thread* communication and synchronization which have been studied: monitors or semaphores and shared memory.

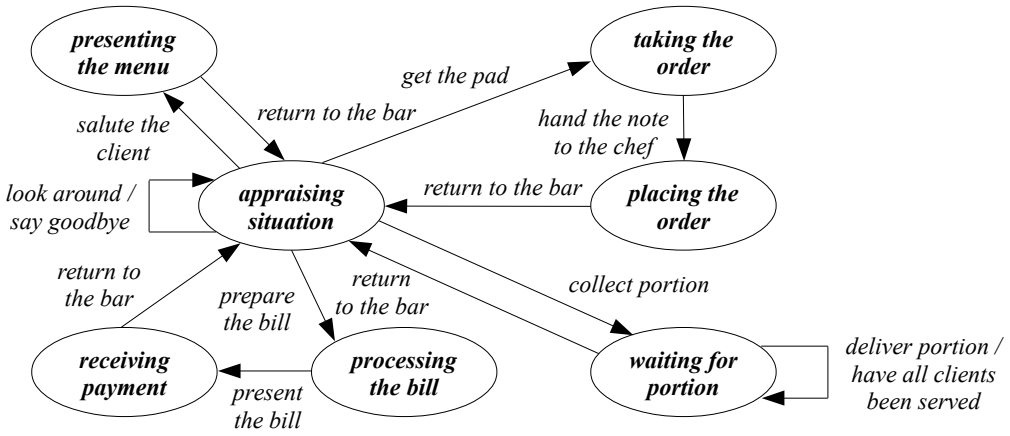
One aims for a distributed solution with multiple information sharing regions, written in Java, run in Linux and which terminates. A *logging* file that describes the evolution of the internal state of the problem in a clear and precise way, must be included.

Suggestion to solution

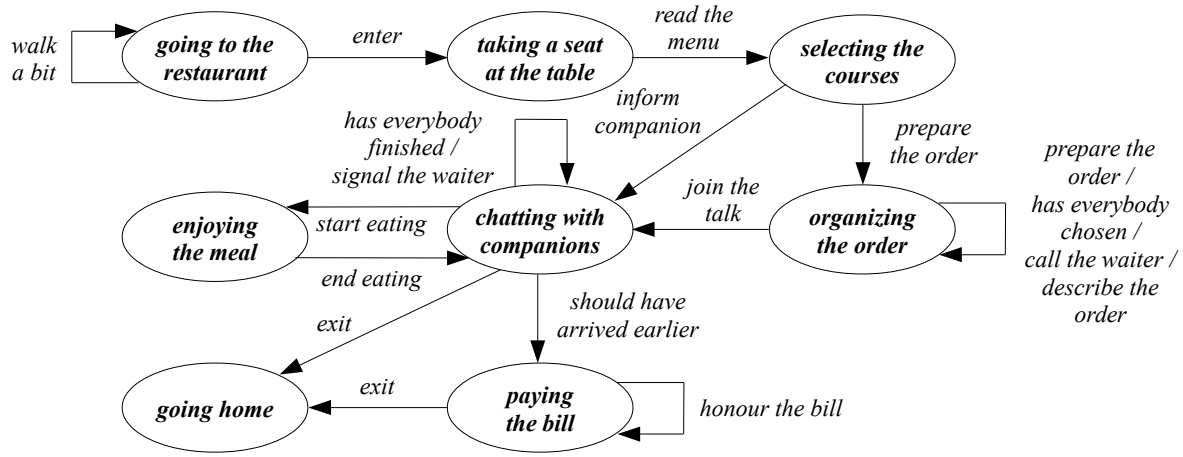
Chef life cycle



Waiter life cycle



Student life cycle



Characterization of the interaction

Chef

WAITING_FOR_AN_ORDER – blocking state (initial state)
the chef is waken up by the operation *handTheNoteToTheChef* of the waiter
PREPARING_THE_COURSE – transition state
DISHING_THE_PORTIONS – transition state
DELIVERING_THE_PORTIONS – blocking state
the chef is waken up by the operation *collectPortion* of the waiter
CLOSING_SERVICE – final state

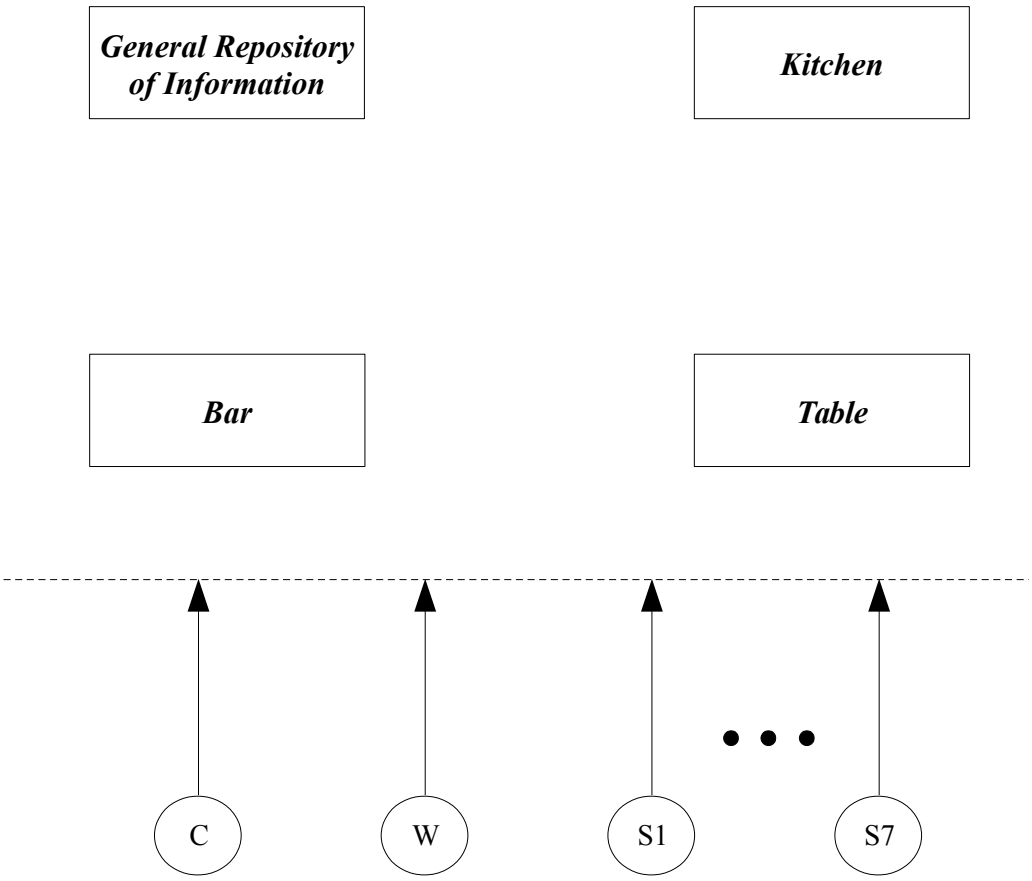
Waiter

APPRAISING_SITUATION – blocking state with transition (initial / final state)
the waiter is waken up by one of the following operations: *alertTheWaiter* of the chef, *enter* and *exit* of all the students, *callTheWaiter* of the first student to sit at the table, *signalTheWaiter* of the last student to finish a course and *shouldHaveArrivedEarlier* of the last student to sit at the table; transition occurs when the last student has left the restaurant
PRESENTING_THE_MENU – blocking state
the waiter is waken up by the operation *readTheMenu* of the student
TAKING_THE_ORDER – blocking state
the waiter is waken up by the operation *describeTheOrder* of the student
PLACING_THE_ORDER – blocking state
the waiter is waken up by the operation *startPreparation* of the chef
WAITING_FOR_PORTION – blocking state
the waiter is waken up by the operation *haveAllPortionsBeenDelivered* of the chef
PROCESSING_THE_BILL – transition state
RECEIVING_PAYMENT – blocking state
the waiter is waken up by the operation *honorTheBill* of the student

Student

GOING_TO_THE_RESTAURANT – transition state with random time (initial state)
TAKING_A_SEAT_AT_THE_TABLE – blocking state
the student is waken up by the operation *saluteTheClient* of the waiter
SELECTING_THE_COURSES – transition state
ORGANIZING_THE_ORDER – blocking state
the student is waken up by the operation *informCompanion* of another student and, when all students are already at the table, by the operation *getThePad* of the waiter
CHATTING_WITH_COMPANIONS – blocking state with transition
the student blocks while waiting for a course to be served and when he/she has finished eating it; transition occurs when the last course has been served and eaten
ENJOYING_THE_MEAL – transition state
PAYING_THE_MEAL – blocking state
the student is waken up by the operation *presentTheBill* of the waiter
GOING_HOME – final state

Information sharing regions



Guidelines for solution implementation

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

2. 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.

3. 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.

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

5. 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.

6. 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.