Continuous Assessment 1

Solution Adopted

Mario Ríos Muñoz
19 February 2014
Computer Science Degree
Advanced Programming Course

Mailbox problem:

We were asked to add a buffer to the mailbox with a capacity of 4 message. In order to implement that I have decided to use an ArrayList of messages (Strings) for the following reasons:

- It is dynamic, so in the case we want to increase the size is easy to change.
- The add method automatically appends an object to the end of the array, so I don't have to care about which was the last message added.
- In order to retrieve a message, I delete the element in the 0 position, returning its value. In this way I make sure that the messages are read in the order they were added (implementing by this some kind of queue).

In order to guaranty mutual exclusion I use synchronized methods, and for making sure that the reader reads only if there is a message i block it while the buffer is empty. Therefore I must notify when a message arrives to unblock the reader.

In the same way, to make sure that the writers don't write when the buffer is full, I block them while this condition happens. Therefore, I must notify when the readers retrieves a message to unblock the writers.

Bar Problem

For this problem I have implemented the following Classes:

- **Person:**
  Has an ID between 1001 and 2000. Is the one who enters in the bar. First decides, then enters if there is enough room, takes a drink and finally exits the pub.
- **Door:**
  Contains a boolean attribute that says wether the door is opened or not. When a person is going to take a drink, enter or exit of the pub, checks if the door is closed. In that case, it waits until the door gets opened again. This is the way I have to stop and resume the threads.
- **Bar Class:**
  Is the one who stores the Persons. It's methods are enter, leave and update. Enter adds a person to the end of an ArrayList. If the bar is full then it blocks until a person leaves (when a person leaves we must notify).
  Leave removes a person from the beginning of the ArrayList. Since this method is executed by a person, there is no need of checking if the ArrayList is empty (if it were empty there would be no one to call this method).
  The method update traverses the ArrayList getting the Ids of the persons and the print all that information in the jTextField of the GUI.