UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia

Control 2. 2015/2016
Ing. Computadores
Ing. Informática
Ing. Software

APELLIDOS _____ NOMBRE _____
DNI _____ ORDENADOR _____ GRUPO _____

1. Semáforos binarios

Supón que un restaurante japonés tiene una mesa con 5 sillas. Cuando un cliente llega al restaurante, si encuentra una silla libre, la ocupa y se sienta a comer. Sin embargo, si un cliente llega al restaurante y todas las sillas están ocupadas, eso significa que todos los clientes que están sentados van en grupo, así que el nuevo cliente que llega (y todos los demás que lleguen después), tiene que esperar a que todo el grupo se haya marchado del restaurante para sentarse.

Implementa este sistema utilizando **semáforos binarios** para asegurar la sincronización entre los procesos. El esqueleto de la solución se encuentra en el campus virtual.

La clase Bar contiene dos métodos:

a) El método ***public void*** *pidoMesa****(int*** *id****)*** es utilizado por el cliente **id** cuando quiere entrar en el restaurante. Si hay sitio, se sienta. Si está lleno, debe esperar a que **TODO** el grupo que ocupa el restaurante se haya marchado antes de sentarse.
b) El método ***public void*** *meVoy****(int*** *id****)*** es utilizado por el cliente **id** cuando se marcha del restaurante

2.- Métodos sincronizados/Locks

Supongamos que hay **n** pasajeros business, y **n** pasajeros normales que desean repetidamente darse una vuelta en un coche de carreras con **C (C<2*n)** plazas, que circula por un circuito. Cuando un pasajero decide darse una vuelta en el coche, espera a que haya una plaza libre para él, teniendo en cuenta que los pasajeros business tienen preferencia sobre los normales, es decir, los normales sólo se suben si no hay pasajeros business esperando a subir. El último pasajero en subir actúa como el conductor del coche, es decir, lo pone en marcha, espera T segundos (por ejemplo, 3) a que se termine la vuelta, y a continuación avisa al resto de los pasajeros del coche para que se bajen. Después de dar una vuelta en el coche, cada pasajero se toma un refresco durante un tiempo antes de volver al circuito para darse otra vuelta.

Implementa este sistema utilizando métodos sincronizados o locks para sincronizar a los procesos. El esqueleto del sistema, que está en el cv, contiene una clase Coche que proporciona los tres métodos siguientes:

a) El método ***public void*** *subirCocheB****(int*** *id****)*** es llamado por el pasajero business **id** cuando quiere darse un paseo en el coche. El pasajero espera hasta que pueda ocupar un asiento en el coche. Cuando se ha sentado, si es el último pasajero en ocupar el coche, lo pone en marcha, espera que finalice la vuelta, y avisa al resto de los pasajeros para que se bajen del coche. Si no es el último, espera sentado hasta que le avisen de que se ha terminado el viaje.

b) El método ***public void*** *subirCocheN(**int** id)* es llamado por el pasajero normal **id** cuando quiere darse un paseo en el coche. El comportamiento es el mismo que el método anterior, salvo que estos procesos tienen menos prioridad para sentarse que los business

c) El método ***private void*** *cocheEnMarcha(**int** id)* llamado por el último pasajero que se monta en el coche, desde alguno de los métodos anteriores. En este método, el pasajero pone en marcha el coche, espera 3 segundos (el tiempo del viaje), y avisa al resto de pasajeros que pueden bajarse del coche.