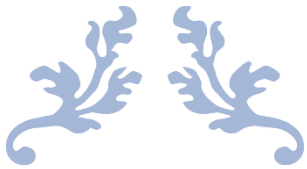

TRABAJO FINAL

Programación de sistemas concurrentes y distribuidos

JORGE LUCAS FERRER
839316

ÍNDICE

Planteamiento del problema

Enunciado:

Se pretende simular el funcionamiento de un gimnasio. Para el acceso al gimnasio, se diferenciará entre la gente que llegue a pie y la gente que llegue en coche. Para los que lleguen con coche, será necesario hacer uso de una aplicación para buscar sitio en el aparcamiento. Esta aplicación solo podrá ser usada por una persona al mismo tiempo, cuando haya alguien usándola los demás esperar su turno.

Al entrar por la puerta, habrá dos opciones. Ser socio(estar dado de alta) o no ser socio (no estar dado de alta) se tendrán que dar de alta en la recepción donde estará el encargado que, a su vez, es monitor del propio gimnasio. Si la recepción está ocupada esperaran hasta que sea su turno.

Si el cliente ya está dado de alta, pasará a la sala de máquinas donde hará su rutina. En la sala de máquinas, habrá un monitor (el de la recepción) y una máquina para cada ejercicio(5 en total). El monitor se dedicará a dar de alta a la gente nueva y, si no hay nadie para darse de alta, responderá dudas a los que están entrenado.

Por último, cuando un cliente acabe en las maquinas decidirá si quiere ducharse o no. Si se ducha ira al vestuario y esperará hasta que una ducha de las tres que hay quede libre, cuando acabe o si elige no ducharse sé irá a casa.

Visto el enunciado, vamos a comentar brevemente como he planteado el ejercicio. Como es normal en este tipo de trabajo y para facilitarlo, se ha dividido el ejercicio en tres partes: la entrada al gimnasio, dentro del gimnasio y salida del gimnasio. Cada parte se compone de otras subpartes.

La entrada, se ocupará de dividir a los clientes entre los que vienen con coche y los que no. Además, también controla que la aplicación funcione correctamente y que en el aparcamiento no se sobrepase el aforo máximo.

La segunda parte, dentro del gimnasio, maneja la organización de las máquinas y la comprobación de que los clientes son socios o no. En caso de que no lo sean y del que monitor este libre, les da de baja.

Por último falta la salida del gimnasio, al salir solo se deberá comprobar si los clientes quieren ducharse o no. En caso afirmativo, esperaran a que quede una ducha libre, se ducharan y saldrán del gimnasio. En caso negativo, saldrán directamente. Finalmente, si el cliente había aparcado, recogerá su coche y se marchará a casa.

Mecanismos elegidos para resolverlo: en qué consisten y por qué se han elegido

Para comentar los mecanismos vamos a diferenciar entre los usados en java y los usados en Ada.

En java, se ha llevado a cabo el ejercicio con el uso de semáforos. En una clase propia y para llevar acabo toda la organización de los semáforos, se han juntado todas las funciones y procedimientos necesarios para simular el "recorrido" que haría un cliente en el gimnasio. En esta clase, no solo se ha hecho uso de los semáforos si no que, también han sido de ayuda los procedimientos "synchronized", propios de los monitores, para facilitar la exclusión mutua en ciertos procedimientos.

Se han elegido estos mecanismos debido a su capacidad para facilitar las necesidades básicas de un programa concurrente, es decir, la exclusión mutua, la inanición,… Por ejemplo, al hacer uso de semáforos no binarios, se asegura que el programa sea altamente equitativo, por lo que no se producirá inanición y, con los semáforos binarios o con los procesos synchronized, será sencillo conseguir exclusión mutua.

En Ada, ha resultado más sencillo si cabe. Ya que, en Ada, los semáforos no están implementados, por lo que, se ha optado por hacer uso de las citas o comunicación directa que nos facilita el lenguaje. Gracias a las citas, se consigue la misma equitatividad y exclusión mutua que con los semáforos en java. Además, como las citas están atentas a dos o más tareas, partes como la del monitor, que resuelve dudas y da de alta, se hacen tremendamente más sencillas.

Resolución del problema.

En java:

Como acabamos de explicar, en java hemos hecho uso de semáforos y de procesos sincronizados. Más allá de los semáforos, el programa se compone de tres clases y la principal.

En la principal o "main", solo se declaran las variables necesarias y se inicializan los procesos clientes. En la clase "monitor", se lleva el movimiento del monitor.En el caso de la clase "clientes", simplemente se hace uso en el orden específico de los métodos que se declaran en la clase "recorrido". Dicha clase "recorrido" está compuesta por todos los métodos (procedimientos y funciones) necesarios para poder llevar a cabo el correcto funcionamiento del gimnasio. Cabe decir que es la clase donde se agrupan los semáforos (se podría haber llamado monitor pero, al haber un monitor del gimnasio iba a ser confuso). Como está clase es la más extensa y la más relevante, vamos a comentarla con un poco más de detalle. La verdad que muchos de los procedimientos son, de una forma u otra, parecidos, así que, explicaré los que para mi son los más interesantes. El primero que quiero comentar es el llamado "monitorRequest", este proceso se encarga de manejar las solicitudes que tiene el monitor, es decir, si algún cliente se tiene que dar de alta o si tiene alguna duda, recurre a este procedimiento para comunicárselo al monitor. Otro procedimiento interesante es "rutina", en este caso me parece interesante porque creo que, debido a mi poca experiencia con este lenguaje, es un tanto redundante y se podría optimizar, ya que, es una repetición del código para cada ejercicio. Aunque crea que este proceso podría estar más optimizado, consiste en un bucle que no para hasta que el cliente realice todas los ejercicios propuestos.

En Ada:

A diferencia de java, en Ada tenemos un único archivo y es el main. Gracias a que en Ada se trabaja a partir de tareas, podemos usar el main para elaborar el trabajo al completo. Como he comentado antes, hemos usado citas para comunicar tareas entre sí. En este caso, los diferentes procesos y tareas se parecen menos que en java pero, como son bastantes, voy a optar por explicar los que a mi parecer son más interesantes, y estos son: espera o monitor (por su parecido podemos englobarlos), rutina y clientes.

Espera o monitor: Quiero explicar estos dos juntos por que se parecen bastante pero, tienen una pequeña diferencia bastante notoria. En el caso de "monitor", se compone por dos "accepts": uno para manejar el mostrador ,dar de alta a los clientes, y otro para resolver las dudas de los clientes. Para que el monitor pueda corresponder a todas las llamadas se hace uso de un "select" y, así, poder seleccionar entre ambos "accepts". En el caso de "espera", las únicas diferencias que encontramos son los usos de los "accepts"(usados para hacer esperar a los procesos clientes en ciertas ocasiones) y la aparición del "when". En esta tarea se usa el when para proponer una condición, si se acepta esta condición se accede al accept por lo que el cliente deja de esperar y, si no se acepta deja al cliente esperando.

Rutina: Este proceso no tiene mucha complicidad pero, en comparación con el procedimiento de java antes comentado, se simplifica mucho gracias a los parámetros de entrada y salida y el procedimiento llamado "ejercicio". En este procedimiento se hace una llamada a la tarea espera para poder llevar un control sobre las maquinas.

Clientes: La tarea clientes es el esquema del recorrido que recorre el cliente en el gimnasio, desde la entrada hasta la salida. Mediante condicionales, booleanos aleatorios y procedimientos previamente declarados, como rutina, completa todos los pasos que debe de hacer cada cliente. Algo remarcable dentro de la tarea es que en un momento llama la tarea espera, mas concretamente al "accept" ducha.

En conclusión y para ambos lenguajes, se ha escogido estos mecanismos por la capacidad de satisfacer las necesidades básicas de la concurrencia(exclusión mutua, ausencia de interbloqueos, ausencia de inanición).

.

Explicación de las diferencias entre las soluciones desarrolladas con los dos mecanismos

Aunque con lo que ya he explicado durante toda la memoria se puede entender las diferencias entre los mecanismos, he de decir que, manteniendo las distancias, no existen tantas diferencias entre los dos mecanismos como cabría esperar. El funcionamiento de los semáforos y las citas tienes sus diferencias como que las citas, necesitan de dos tareas como mínimo para funcionar, una para emitir el accept y la otra para invocarla y, en el caso de los semáforos se pueden comunicar de un proceso a otro o en el mismo varias veces. Otra diferencia, la encontramos en que, en las citas, al llamar a un accept este ejecutara el código dentro de si mismo pero, en los semáforos, simplemente servirán como parada dentro del código, no añadirán nada más. Pero el proceso lógico que siguen para mi es ciertamente parecido.