Programación Concurrente

Práctica 3: Monitores

1. Considere el siguiente monitor

```
Monitor m{
   condicion permiso

   public void antesydespues {
        //antes
        permiso.signal()
       //despues   }

   public void parte2{
       permiso.wait()
       //importante }
 }
```

Este monitor fue diseñado para que la parte importante sea ejecutada por un thread, y las partes antes y después sean ejecutadas por otro thread y además se asegure que el único órden de ejecución posible sea:

```
antes; importante; despues
```

   a) Mostrar que el código del monitor no es correcto si la disciplina utilizada es "signal y espera urgente". Justifique su respuesta mostrando una traza.

   b) ¿Qué sucede si se utiliza la disciplina"signal y continúa"?

2. Un secuenciador ternario provee tres operaciones `primero`, `segundo`, `tercero`. Se desea implementar usando monitores al `secuenciador ternario` para coordinar a threads que pueden invocar a cualquiera de las operaciones. El secuenciador alternará cíclicamente la ejecución de `primero`, luego de `segundo`, y finalmente `tercero`.

3. Se desea implementar usando monitores una barrera para coordinar a N threads. Una barrera provee una única operación denominada esperar. La idea es que cada uno de los N threads invocarán una vez a la operación esperar y el efecto de invocarla es que el thread se bloquea y no puede continuar hasta tanto los restantes threads invoquen a la operación esperar. Por ejemplo, si mibarrera es una barrera para coordinar 3 threads, el uso de mibarrera en los siguientes threads

```
thread1: print 'a'; mibarrera.esperar(); print 1
thread2: print 'b'; mibarrera.esperar(); print 2
thread3: print 'c'; mibarrera.esperar(); print 3
```

garantiza que todas las letras se mostrarán antes de los números. Dar una implementación para el monitor barrera.

4. Se desea implementar usando monitores un `atrapador` que permite coordinar varios threads. Un atrapador provee dos operaciones `esperar` y `liberar (N)`. La idea es que cada uno de los threads puede invocar a cualquiera de las operaciones: un thread que invoca a `esperar` se deberá bloquear hasta tanto sea liberado por algún otro thread que invoca a la operación $liberar(N)$. El parámetro N indica cuantos threads de los que están bloqueados deberán liberarse.

   a) Dar una solución en la que el thread que invoca $liberar(N)$ nunca se bloquea. Además sólo se liberan threads si hay N o más threads esperando. En el caso en que haya menos de N threads, ninguno se liberará y deberán continuar esperando.

   b) Modificar la solución anterior para hacer que en el caso en que haya menos de N threads esperando, el proceso que invoca a liberar se bloquee hasta tanto se puedan liberar N threads (todos los threads que esperan se deben liberar juntos).

5. Se desea resolver utilizando monitores el problema de la peluquería en la cual los clientes y los peluqueros son modelados como threads independientes que se sincronizan utilizando un monitor, denominado pelu, que tiene la siguientes operaciones:

   - cortarseElPelo
   - empezarCorte
   - terminarCorte

   Los threads peluqueros se comportan de la siguiente manera

   ```
   while(true){
       pelu.empezarCorte()
       //cortar
       pelu.terminarCorte()
   }
   ```

   mientras que los clientes invocan a la operacóin `cortarseElPelo`. El funcionamiento es el esperado para una peluquería. Los clientes llegan y esperan hasta ser atendidos. Solo se van de la peluquería cuando finalizan de cortarles el pelo. Se solicita:

   a) Dar una implementación a este problema utilizando monitores (con la política signal y continua y utilizando diferentes variables de condición).

   b) Cómo se modifica su solución si puede utilizar una única variable de condición.

6. El comité organizador de una conferencia posee una sala para exposición de charlas de distintos temas. Las personas que desean asistir a una charla entran en la sala y esperan hasta que la charla comienza. Las charlas empiezan cuando el orador llega a la sala. Por respeto, los asistentes no se retiran hasta que la charla termina ni entran cuando la charla ya ha comenzado. La sala posee una capacidad de 50 personas, sin contar al orador. De una solución usando monitores que modele los siguientes escenarios:

   a) Se dispone de una sala para dar repetidas veces la misma charla. El orador descansa 5 minutos entre charla y charla. Si al momento de iniciar la charla el auditorio está vacío el orador descansa otros 5 minutos esperando que lleguen oyentes. Cuando el auditorio se llena los asistentes deben esperar a que comience la siguiente charla luego del descanso del orador.

   b) Al igual que en el punto a) se dispone de una sala para dar tres charlas distintas, pero en este caso los oradores esperan a que haya al menos 40 personas en el auditorio.

7. Se desea implementar utilizando monitores la siguiente sala de apuestas. La sala de apuestas propone un acertijo (básicamente una palabra que los apostadores deben descubrir). Los apostadores proponen una palabra y un monto. Si aciertan ganan 10 veces su apuesta y el acertijo concluye. Si un participante falla, no puede volver a apostar hasta tanto haya participado otro jugador. Al final, cada apostador debe mostrar por pantalla si ganó o no y el monto. El monitor *Juego* que modela a la sala de apuestas debe proveer las siguientes operaciones.

   - *concluido*: responde si el acertijo finalizó o no.
   - *apostar(palabra,apuesta)* con la cual un jugador realiza su apuesta. Retorna verdadero si el jugador ganó su apuesta y falso en caso contrario.

   Se solicita:

   a) Dar la implementación de un jugador (asumiendo que se pasa la sala de juegos como parámero al momento de la construcción). El jugador no puede asumir que el monitor implementa operaciones distintas de las mencionadas anteriormente.

   b) Dar la implementación del monitor *Juego* que garantice que ningún Jugador que se comporte según el código definido anteriormente pueda apostar dos veces consecutivas y que finaliza cuando el acertijo concluye (aún si el que adivina es otro jugador).

8. En el bar X se fabrican dos tipos de pizzas: las chicas y las grandes. El maestro pizzero va colocando de a una las pizzas listas en una barra para que los clientes se sirvan y pasen luego por la caja. Lamentablemente todos los clientes tienen buen apetito y se comportan de la siguiente manera: prefieren siempre tomar un pizza grande, pero si no lo logran se conforman con dos pequeñas. Los clientes son mal educados y no esperan en una cola (todos compiten por las pizzas). Dar una solución en donde cada cliente es modelado como un thread. Como mecanismo de sincronización, utilizar monitores.

9. Considerar el siguiente problema: Se desea modelar un bote utilizado por personas para cruzar un río. En todo momento el bote se encuentra en una de las dos costas del río (norte o sur). Las personas que llegan a una de las costas pueden abordar el bote si el mismo se encuentra en esa orilla y aún tiene lugar disponible (el bote tiene una capacidad fija que se define al momento de construcción). El bote puede atravesar el río sólo cuando tiene una autorización y cuando está completo. Cuando llega a la orilla, descarga a todas las personas y carga a las que están esperando. Cuando se llene y vuelva a tener autorización, volverá a cruzar el río en dirección opuesta. Notar que la autorización puede llegar antes o después de que esté completo.

   a) Modelar el problema utilizando monitores

   b) Su modelo contempla el hecho de que las personas que esperan deberían ingresar en el orden en que arribaron? Cómo modificaría la solución para considerar además esta restricción?.

10. Se desea implementar un monitor que permita administrar la asignación de recursos. Esta clase mantiene una cola de recursos de un determinado tipo y permite que se tomen y liberen recursos a través de dos operaciones tomar (que devuelve el primer elemento de la cola) y liberar (que agrega a la cola el elemento que se libera).

    a) Dar una implementación en Java.

    b) Modificar la solución anterior de modo tal que un proceso pueda tomar y liberar una cantidad arbitraria de recursos al mismo tiempo.

    c) Dar una solución a este problema de modo tal que no se perjudique a aquellos que solicitan una gran cantidad de recursos.