EJERCICIOS DE CONCURRENCIA

A continuación se muestra un conjunto de problemas en los que diferentes hilos o tareas trabajan de forma concurrente. Se requiere proporcionar una solución para estos problemas, tanto en pseudo-código (por ejemplo, utilizando el concepto general de monitor), como en Java (utilizando las primitivas básicas de java o bien las herramientas de la librería *java.util.concurrent*). Compruebe que su implementación en Java es correcta, realizando diferentes ejecuciones y asegurándose de que no se produzcan condiciones de carrera (ni se puedan llegar a producir nunca) y que se cumplan todas las condiciones de sincronización.

Procesos en equilibrio

En un sistema concurrente existen dos tipos de procesos, A y B, que compiten por utilizar cierto recurso. Al recurso se accede mediante la rutina de solicitarlo, esperar hasta que se pueda usar, usarlo y luego liberarlo.

En cualquier momento puede haber un máximo de N procesos de cualquier tipo usando el recurso (N es constante). Por otro lado, para que un proceso de tipo A pueda entrar a emplear el recurso, debe haber al menos el doble de procesos de tipo B que procesos de tipo A dentro del recurso.

Los fumadores

En torno a una mesa hay cuatro personas: tres fumadores y un proveedor. Cada fumador fuma cigarrillos uno detrás de otro. Para poder fumarse un cigarrillo se necesitan tres ingredientes: papel, tabaco y fósforos. Uno de los fumadores tiene papel, el otro tabaco y el otro fósforos. El proveedor tiene una cantidad infinita de los tres ingredientes.

Inicialmente, el agente coloca dos de los ingredientes en la mesa, escogidos al azar. El fumador que tiene el ingrediente que falta toma lo que hay en la mesa, lía un cigarrillo y se pone a fumar. El proveedor inmediatamente toma dos ingredientes y los coloca en la mesa. De esta forma, mientras un fumador está fumando, se da la posibilidad de que otro fumador pueda tomar la pareja de ingredientes que hay en la mesa y así lleguemos a tener varios fumadores al mismo tiempo. (Aunque esto no ocurrirá si el proveedor coloca justo el par de ingredientes que no le sirven a los fumadores en espera, pero eso depende del azar).

El taller de costura

Tenemos un taller de costura dedicado a hacer jerséis. En su interior tenemos a tres personas trabajando. Una persona está continuamente fabricando mangas, que va depositando en un cesto. El cesto tiene una capacidad limitada: cuando se llena, la costurera deja de coser más mangas hasta que hay hueco libre. Otra persona está

continuamente fabricando los cuerpos de los jerséis, que también deposita en su correspondiente cesta de capacidad limitada. Una tercera persona se encarga continuamente de ensamblar jerséis, cogiendo en cada caso dos mangas de la cesta de mangas y un cuerpo de la cesta de cuerpos.

Se trata de escribir el código que sincronice a estas tres personas, de forma que las dos primeras personas no avancen si su cesta está llena, y que la tercera persona no avance si le faltan piezas para hacer un nuevo jersey. Se supone que las capacidades de las dos cestas son constantes y distintas (supongan, p.ej. que son dos constantes enteras "NumMaxMangas" y "NumMaxCuerpos").

El barbero dormilón (the sleeping barber)

Este problema fue planteado por Edsger Dijkstra. En una barbería, donde trabaja un barbero, se dispone de una sala de espera con N sillas y una silla de barbero (donde se sienta el cliente para que le corten el pelo). Si no hay clientes, el barbero se echa a dormir. Si un cliente entra en la barbería y encuentra todas las sillas ocupadas, se marcha de la barbería. Si el barbero está ocupado, pero hay sillas disponibles, el cliente se sienta en una de ellas. Si el barbero está dormido, el cliente lo despierta y se sienta en la silla del barbero, para que le corten el pelo. Cuando el barbero termina de atender a un cliente, va a la sala de espera y, si hay clientes esperando, se lleva a uno de ellos a la silla del barbero y comienza a atenderle. Si no hay clientes, se echa a dormir (en la silla del barbero).

Se trata de escribir un programa que coordine al barbero y los clientes, cada uno de los cuales se corresponde con un hilo.

El puente estrecho

Construya un sistema para gestionar el tráfico sobre un puente estrecho. El puente lo pueden atravesar vehículos desde ambos extremos. Como el puente tiene un solo carril, en un momento dado sólo puede haber vehículos cruzándolo en un único sentido. El puente tiene una capacidad de carga limitada: si hay más de tres vehículos, se hunde. Implemente una solución en la que cada vehículo sea un hilo. El algoritmo que sigue cada vehículo debe ser parecido a esto:

```
void vehículo (sentido) {
       llegar_al_puente(sentido)
       cruzar_puente(sentido)
       salir_del_puente(sentido)
}
```

El parámetro *sentido* indica en qué sentido se cruza el puente (o sea, que puede tener dos valores, por ejemplo *izquierda* y *derecha).*

Garantice que no ocurren colisiones y que el puente no se hunde. Además, intente que no se produzcan esperas indefinidas.

Problema del baño mixto

Ante la carencia de baños en el Departamento de Sistemas Informáticos y Computación, el conserje ha decidido que los pocos baños existentes sean de uso mixto: pueden entrar tanto hombres como mujeres, pero con la condición de que simultáneamente sólo pueda haber personas de un único sexo. Además, el baño tiene una capacidad limitada.

Implemente la gestión de uno de estos baños (cada persona se simula como un hilo que entra y sale del baño). Su solución debe estar libre de inanición.

Caníbales y misioneros

En el África profunda viven junto a un río una tribu de caníbales y un grupo de misioneros. El río se puede cruzar mediante un bote, con capacidad para tres personas. Como las aguas son bravas, el bote siempre tiene que transportar carga completa. Los misioneros no pueden ser superados en número por los caníbales, pues si no, los caníbales se comerían a los misioneros.

El sistema debe lanzar varios hilos, uno por cada persona. Escriba dos procedimientos: LlegaMisionero() y LlegaCaníbal(), para cuando cada persona llega al bote. Estos procedimientos dejan al hilo esperando hasta que el bote se encuentre lleno y seguro. Cuando esto ocurra, el procedimiento retorna y se considera que el barco vuelve a quedar vacío.