UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

Programación avanzada II

Laboratorio 6-Semáforos

Los esqueletos para realizar los ejercicios están en el campus virtual.

1- En un sistema industrial existen tres sensores que realizan mediciones del nivel de temperatura, humedad y luz respectivamente. Cuando se han recogido mediciones de los tres sensores, existe un dispositivo "trabajador" encargado de realizar ciertas tareas según las mediciones realizadas.

El dispositivo **no puede** comenzar a realizar sus tareas hasta que se han recogido mediciones de los tres sensores, y los sensores **no pueden** volver a realizar mediciones hasta que el dispositivo finaliza sus tareas. El proceso se repite de forma indefinida de manera que cuando el dispositivo finaliza sus tareas, volverá a esperar a que haya mediciones de los tres sensores.

Realiza, utilizando **semáforos binarios,** el modelado de dicho sistema. Modela el dispositivo trabajador y cada sensor como una hebra (con lo cual habrá un total de 4 hebras). Modela el proceso de realizar mediciones y las tareas del dispositivo con retrasos aleatorios y valores de tipo entero. Inicialmente puede suponerse que los sensores pueden comenzar haciendo peticiones. Un ejemplo de ejecución de este sistema es:

```
Thread-2: Sensor 2 almacena su medición
Thread-0: Sensor 0 almacena su medición
Thread-1: Sensor 1 almacena su medición
Thread-3: El trabajador recoge las mediciones
Thread-3: El trabajador ha terminado sus tareas
Thread-1: Sensor 1 almacena su medición
Thread-0: Sensor 0 almacena su medición
Thread-2: Sensor 2 almacena su medición
Thread-3: El trabajador recoge las mediciones
Thread-3: El trabajador ha terminado sus tareas
...
```

2- En una cadena de montaje existe un robot encargado de colocar productos de 3 tipos diferentes (0, 1 o 2) en la cadena de montaje. Otros robots, retiran los productos de la cadena de montaje para realizar su empaquetado, teniendo en cuenta que están especializados en un solo tipo de producto (0, 1 o 2), ignorando los que no son de su tipo. Finalmente, se quiere llevar un control del total de productos empaquetados (independientemente de su tipo). Modelar utilizando **semáforos binarios** el sistema descrito con las siguientes indicaciones:

- Modela cada robot como una hebra (1 colocador y 3 empaquetadores, uno para cada tipo de producto).
- Los productos son colocados de uno en uno en la cadena, y solamente en posiciones libres (se puede considerar que en la cadena de montaje caben un máximo N de elementos). Si no hay posiciones libres el robot colocador tendrá que esperar hasta que algún producto sea retirado de la cadena.
- Los robots empaquetadores se especializan en un tipo de producto (0, 1 o 2) en tiempo de inicialización.
- Los robots empaquetadores comprueban si hay algún elemento de su tipo en la cadena ignorando los productos que no sean de su tipo. Si hay algún producto de su tipo lo retiran de la cadena (sólo 1 producto cada vez) y la posición queda libre para colocar nuevos productos, en caso contrario se quedan a la espera de que haya nuevos productos.
- Los robots empaquetadores de distinto tipo pueden funcionar a la vez.

- Tanto el colocador como los empaquetadores nunca acaban.
- Cada vez que un robot empaquetador procesa un producto, la cuenta total de productos empaquetados debe aumentar y mostrarse un mensaje por pantalla.

Un ejemplo de ejecución de este sistema para un buffer de tamaño 5 podría ser (en el buffer se almacena sólo el número de elementos de cada tipo que contiene):

```
Thread-3: Colocador pone un producto 2. Quedan [0,0,1]
Thread-3: Total de productos empaquetados 1
Thread-2: Empaquetador 2 retira un producto. Quedan [0,0,0]
Thread-3: Colocador pone un producto 1. Quedan [0,1,0]
Thread-3: Total de productos empaquetados 2
Thread-1: Empaquetador 1 retira un producto. Quedan [0,0,0]
Thread-3: Colocador pone un producto 2. Quedan [0,0,1]
Thread-3: Total de productos empaquetados 3
Thread-3: Colocador pone un producto 2. Quedan [0,0,2]
Thread-3: Total de productos empaquetados 4
Thread-2: Empaquetador 2 retira un producto. Quedan [0,0,1]
Thread-3: Colocador pone un producto 1. Quedan [0,1,1]
Thread-3: Total de productos empaquetados 5
Thread-3: Colocador pone un producto 1. Quedan [0,2,1]
Thread-3: Total de productos empaquetados 6
Thread-3: Colocador pone un producto 2. Quedan [0,2,2]
Thread-3: Total de productos empaquetados 7
Thread-1: Empaquetador 1 retira un producto. Quedan [0,1,2]
Thread-3: Colocador pone un producto 1. Quedan [0,2,2]
Thread-3: Total de productos empaquetados 8
Thread-3: Colocador pone un producto 0. Quedan [1,2,2]
Thread-3: Total de productos empaquetados 9
Thread-0: Empaquetador 0 retira un producto. Quedan [0,2,2]
Thread-2: Empaquetador 2 retira un producto. Quedan [0,2,1]
Thread-3: Colocador pone un producto 0. Quedan [1,2,1]
Thread-3: Total de productos empaquetados 10
Thread-3: Colocador pone un producto 0. Quedan [2,2,1]
Thread-3: Total de productos empaquetados 11
....
```

3- Supón que un Centro Comercial dispone de unos **aseos** suficientemente grandes para dar servicio a sus clientes. Además de los **clientes**, el Centro Comercial tiene un **equipo de limpieza** que periódicamente limpia los aseos. El sistema debe satisfacer las siguientes condiciones de sincronización:

a) Cualquier número de clientes puede estar simultáneamente utilizando los aseos. Se supone que son tan grandes que, cuando un cliente quiere entrar, siempre hay sitio disponible.
b) Mientras el equipo de limpieza trabaja en los aseos, no puede haber ningún cliente dentro.

Por ejemplo, una ejecución del sistema podría ser:

```
Thread-2: Entra cliente 2. Hay 1 clientes.
Thread-1: Entra cliente 1. Hay 2 clientes.
Thread-8: Entra cliente 8. Hay 3 clientes.
Thread-8: Sale cliente 8. Hay 2 clientes.
Thread-9: Entra cliente 9. Hay 3 clientes.
Thread-1: Sale cliente 1. Hay 2 clientes.
Thread-7: Entra cliente 7. Hay 3 clientes.
Thread-2: Sale cliente 2. Hay 2 clientes.
Thread-8: Entra cliente 8. Hay 3 clientes.
Thread-7: Sale cliente 7. Hay 2 clientes.
Thread-8: Sale cliente 8. Hay 1 clientes.
Thread-9: Sale cliente 9. Hay 0 clientes.
Thread-6: Entra cliente 6. Hay 1 clientes.
Thread-6: Sale cliente 6. Hay 0 clientes.
Thread-5: Entra cliente 5. Hay 1 clientes.
Thread-5: Sale cliente 5. Hay 0 clientes.
Thread-10:      Entra el equipo de limpieza.
Thread-10:      Sale el equipo de limpieza.
Thread-9: Entra cliente 9. Hay 1 clientes.
Thread-0: Entra cliente 0. Hay 2 clientes.
Thread-9: Sale cliente 9. Hay 1 clientes.
Thread-8: Entra cliente 8. Hay 2 clientes.
Thread-0: Sale cliente 0. Hay 1 clientes.
Thread-8: Sale cliente 8. Hay 0 clientes.
Thread-10:      Entra el equipo de limpieza.
Thread-10:      Sale el equipo de limpieza.
Thread-1: Entra cliente 1. Hay 1 clientes.
Thread-4: Entra cliente 4. Hay 2 clientes.
....
```

4- **El problema de la montaña rusa**. Supón que hay **n** procesos **pasajeros** y **un** proceso **coche**. Los pasajeros esperan repetidamente para darse una vuelta en el coche, que tiene una capacidad de pasajeros **C<n**. Sin embargo, **el coche sólo da una vuelta cuando está lleno**. El coche tarda T segundos en dar una vuelta, una vez que está lleno. Después de dar una vuelta, cada pasajero da un paseo por el parque de atracciones durante un tiempo aleatorio, antes de volver a la montaña rusa para darse otra vuelta. Diseña un programa que resuelva este problema utilizando sólo **semáforos binarios** para sincronizar las hebras.

Por ejemplo, una ejecución de este sistema con un coche de capacidad C = 5 podría ser:

```
Thread-3: El pasajero 2 se sube al coche. Hay 1 pasajeros.
Thread-6: El pasajero 5 se sube al coche. Hay 2 pasajeros.
Thread-5: El pasajero 4 se sube al coche. Hay 3 pasajeros.
Thread-11: El pasajero 10 se sube al coche. Hay 4 pasajeros.
```

```
Thread-12: El pasajero 11 se sube al coche. Hay 5 pasajeros.
Thread-0:       Coche lleno!!! empieza el viaje....
Thread-0:       Fin del viaje... :-(
Thread-3: El pasajero 2 se baja del coche. Hay 4 pasajeros.
Thread-6: El pasajero 5 se baja del coche. Hay 3 pasajeros.
Thread-5: El pasajero 4 se baja del coche. Hay 2 pasajeros.
Thread-11: El pasajero 10 se baja del coche. Hay 1 pasajeros.
Thread-12: El pasajero 11 se baja del coche. Hay 0 pasajeros.
Thread-2: El pasajero 1 se sube al coche. Hay 1 pasajeros.
Thread-1: El pasajero 0 se sube al coche. Hay 2 pasajeros.
```

```
Thread-4: El pasajero 3 se sube al coche. Hay 3 pasajeros.
Thread-9: El pasajero 8 se sube al coche. Hay 4 pasajeros.
Thread-7: El pasajero 6 se sube al coche. Hay 5 pasajeros.
Thread-0:       Coche lleno!!! empieza el viaje....
Thread-0:       Fin del viaje... :-(


....
```

5- Supón que átomos de **hidrógeno** y **oxígeno** están dando vueltas en el espacio, intentando agruparse para formar moléculas de agua. Para ello es necesario que dos átomos de hidrógeno y uno de oxígeno se sincronicen. Supongamos que cada átomo de hidrógeno y oxígeno está simulado por un proceso. La gestión de la sincronización de los átomos tiene lugar en un objeto gestor de la clase **GestorAgua**. Cada átomo de hidrógeno llama al método hListo cuando quiere formar parte de una molécula. Del mismo modo los átomos de oxígeno llaman a oListo cuando quieren combinarse con dos hidrógenos para formar agua. Los procesos deben esperar en estos métodos hasta que sea posible formar la molécula. Implementa una solución utilizando semáforos **binarios** que resuelva este problema. Por ejemplo, una ejecución de este programa con 10 átomos de hidrógeno y 5 de oxígeno podría ser:

```
Thread-4: Hidrógeno 4 quiere formar una molécula
Thread-0: Hidrógeno 0 quiere formar una molécula
Thread-12: Oxígeno 2 quiere formar una molécula
Thread-12:      Molécula formada!!!
Thread-6: Hidrógeno 6 quiere formar una molécula
Thread-9: Hidrógeno 9 quiere formar una molécula
Thread-14: Oxígeno 4 quiere formar una molécula
Thread-14:      Molécula formada!!!
Thread-5: Hidrógeno 5 quiere formar una molécula
Thread-13: Oxígeno 3 quiere formar una molécula
Thread-1: Hidrógeno 1 quiere formar una molécula
```

```
Thread-1:       Molécula formada!!!
Thread-11: Oxígeno 1 quiere formar una molécula
Thread-3: Hidrógeno 3 quiere formar una molécula
Thread-2: Hidrógeno 2 quiere formar una molécula
Thread-2:       Molécula formada!!!
Thread-10: Oxígeno 0 quiere formar una molécula
Thread-8: Hidrógeno 8 quiere formar una molécula
Thread-7: Hidrógeno 7 quiere formar una molécula
Thread-7:       Molécula formada!!!
main: Fin del programa
```

6- Considera un sistema formado por tres hebras fumadores que se pasan el día liando cigarros y fumando. Para liar un cigarro necesitan tres ingredientes: tabaco, papel y cerillas. Cada fumador dispone de un surtido suficiente (para el resto de su vida) de uno de los tres ingredientes. Cada fumador tiene un ingrediente diferente, es decir, un **fumador** tiene una cantidad infinita de tabaco, el otro de papel y el otro de cerillas. Hay también una hebra **agente** que pone dos de los tres ingredientes encima de una **mesa**. El agente dispone de unas reservas infinitas de cada uno de los tres ingredientes y escoge de forma aleatoria cuáles son los ingredientes que pondrá encima de la mesa. Cuando los ha puesto, el fumador que tiene el otro ingrediente puede fumar (los otros dos no). Para ello coge los ingredientes, se lía un cigarro y se lo fuma. Cuando termina de fumar vuelve a repetirse el ciclo. En resumen, el ciclo que debe repetirse es:
"agente pone ingredientes → fumador hace cigarro → fumador fuma → fumador termina de fumar → agente pone ingredientes → ...

Un ejemplo de ejecución de este sistema es:

```
Thread-3: El agente no pone ingrediente 0
Thread-0: Fumador 0 fuma
Thread-0: Fumador 0 termina de fumar
Thread-3: El agente no pone ingrediente 2
Thread-2: Fumador 2 fuma
```

```
Thread-2: Fumador 2 termina de fumar
Thread-3: El agente no pone ingrediente 2
Thread-2: Fumador 2 fuma
Thread-2: Fumador 2 termina de fumar
Thread-3: El agente no pone ingrediente 0
```

```
Thread-0: Fumador 0 fuma
Thread-0: Fumador 0 termina de fumar
Thread-3: El agente no pone ingrediente 2
Thread-2: Fumador 2 fuma
Thread-2: Fumador 2 termina de fumar
Thread-3: El agente no pone ingrediente 1
Thread-1: Fumador 1 fuma
Thread-1: Fumador 1 termina de fumar
Thread-3: El agente no pone ingrediente 2
Thread-2: Fumador 2 fuma
Thread-2: Fumador 2 termina de fumar
...
```

7- Considera un nido con **n pájaros bebés** y **dos pájaros padres** (el papá y la mamá). Todos los pájaros comparten un plato común que puede contener a lo suma **B** bichitos. Cada pájaro padre (papá o mamá) da una vuelta volando, atrapa un bichito, vuelve al nido, espera a que haya sitio en el plato, deposita en él el bichito capturado, y repite todas las acciones de nuevo. Cada pájaro bebé pía un ratito, espera a que el plato tenga algún bichito, lo coge, se lo come y repite de nuevo todas las acciones. Implementa este sistema utilizando **semáforos binarios**, suponiendo un comportamiento infinito para cada uno de los procesos.

Por ejemplo, una ejecución de este sistema con un plato de capacidad 5 podría ser:

```
Thread-10: Papá 0 pone un bichito. Quedan 1 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 2 bichitos
Thread-9: Bebé 9 coge un bichito. Quedan 1 bichitos
Thread-0: Bebé 0 coge un bichito. Quedan 0 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 1 bichitos
Thread-3: Bebé 3 coge un bichito. Quedan 0 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 1 bichitos
Thread-10: Papá 0 pone un bichito. Quedan 2 bichitos
Thread-4: Bebé 4 coge un bichito. Quedan 1 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 2 bichitos
Thread-10: Papá 0 pone un bichito. Quedan 3 bichitos
Thread-2: Bebé 2 coge un bichito. Quedan 2 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 3 bichitos
Thread-10: Papá 0 pone un bichito. Quedan 4 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 5 bichitos
Thread-8: Bebé 8 coge un bichito. Quedan 4 bichitos
Thread-0: Bebé 0 coge un bichito. Quedan 3 bichitos
Thread-2: Bebé 2 coge un bichito. Quedan 2 bichitos
Thread-7: Bebé 7 coge un bichito. Quedan 1 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 2 bichitos
Thread-10: Papá 0 pone un bichito. Quedan 3 bichitos
Thread-5: Bebé 5 coge un bichito. Quedan 2 bichitos
Thread-11: Papá 1 pone un bichito. Quedan 3 bichitos
Thread-1: Bebé 1 coge un bichito. Quedan 2 bichitos
Thread-10: Papá 0 pone un bichito. Quedan 3 bichitos
Thread-4: Bebé 4 coge un bichito. Quedan 2 bichitos
Thread-9: Bebé 9 coge un bichito. Quedan 1 bichitos
...
```

8- La tribu de los caníbales Ngoro-Ngoro está de enhorabuena. Han capturado una nutrida expedición de exploradores de una conocida revista de divulgación científica. Para celebrarlo, los Ngoro-Ngoro, que no piensan en el futuro, organizan una fiesta en la que toda la tribu baila y come de una **olla** central en la que el cocinero va cocinando a los exploradores capturados. Para comer, cada caníbal se sirve en su propio plato de la olla central. **Cuando un caníbal va a comer y encuentra la olla vacía, llama al cocinero para que vuelva a preparar otro explorador y lo eche a la olla**. Pero preparar un explorador es una tarea bastante cansada, por lo que, mientras la olla no está vacía, el cocinero se retira a su choza a descansar. Diseña una solución basada en **semáforos binarios** que simule la gran fiesta de los Ngoro-Ngoro en la que se evite despertar al cocinero cuando no haga falta. Supón que la comida no se acaba nunca y que cada explorador da lugar a **R** raciones que los caníbales comen de una en una, y que la fiesta comienza cuando la olla está llena con el primer explorador. Un ejemplo de ejecución de este sistema con R = 5 y 20 caníbales podría ser:

```
Thread-12: Caníbal 6 coge una ración de la olla. Quedan 4 raciones.
Thread-24: Caníbal 12 coge una ración de la olla. Quedan 3 raciones.
Thread-22: Caníbal 11 coge una ración de la olla. Quedan 2 raciones.
Thread-8: Caníbal 4 coge una ración de la olla. Quedan 1 raciones.
Thread-24: Caníbal 12 coge una ración de la olla. Quedan 0 raciones.
Thread-1: El cocinero llena la olla. Quedan 5 raciones.
Thread-22: Caníbal 11 coge una ración de la olla. Quedan 4 raciones.
Thread-2: Caníbal 1 coge una ración de la olla. Quedan 3 raciones.
Thread-14: Caníbal 7 coge una ración de la olla. Quedan 2 raciones.
Thread-36: Caníbal 18 coge una ración de la olla. Quedan 1 raciones.
Thread-16: Caníbal 8 coge una ración de la olla. Quedan 0 raciones.
Thread-3: El cocinero llena la olla. Quedan 5 raciones.
Thread-18: Caníbal 9 coge una ración de la olla. Quedan 4 raciones.
Thread-4: Caníbal 2 coge una ración de la olla. Quedan 3 raciones.
Thread-0: Caníbal 0 coge una ración de la olla. Quedan 2 raciones.
Thread-12: Caníbal 6 coge una ración de la olla. Quedan 1 raciones.
Thread-24: Caníbal 12 coge una ración de la olla. Quedan 0 raciones.
Thread-5: El cocinero llena la olla. Quedan 5 raciones.
....
```