Trabajo práctico Nº 4

Mecanismos de sincronización:

Semáforos binarios

1. Dado los siguientes códigos:

a)

```
public class SynchronizedCounter {
        private int c = 0;
        public synchronized void increment() {c++;}
        public  void decrement() {c--;}
        public synchronized int value() {return c;}
    }
```

b)

```
public class SynchronizedObjectCounter {
        private int c = 0;
        public  void increment(){
                synchronized (c) {c++;} // Este elemento debe ser casteado a
Integer
         }
        public  void decrement() {
                 synchronized (this) {c--;}
         }
        public int value() {return c;}
}
```

i. Verifique el funcionamiento del código del inciso a)

ii. Verifique el funcionamiento del código del inciso b)

iii. En caso de ser necesario realice las correcciones que crea convenientes.

c) Compare para este caso, métodos sincronizados con objetos sincronizados. (Ventajas y Desventajas)

2. Dados los siguientes procesos:

considere que sem1, sem2, sem3, sem4 son semáforos binarios.

```
Proceso P1
      adquirir(sem2);
      operaciones_P1;
      liberar(sem1);
```

```
end
```

```
Proceso P2
   adquirir(sem3);
   operaciones_P2;
   liberar(sem2);
end
```

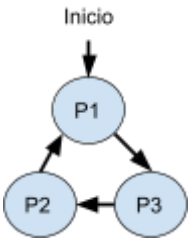
```
Proceso P3
   adquirir(sem1);
   operaciones_P3;
   liberar(sem3);
   liberar(sem4);
end
```

```
Proceso P4
     adquirir(sem4);
     operaciones_P4;
end
```

```
inicializa(sem1,0);
inicializa(sem2,1);
inicializa(sem3,0);
inicializa(sem4,0);
```

- ¿Qué sucede si el semáforo sem2 se inicializa en 0?
- ¿Qué sucede si el semáforo sem1 se inicializa en 1?

3. Implementar el código correspondiente para lograr la sincronización de tres procesos (P1,P2,P3) de forma que se establezca el orden de ejecución P1, P3 y P2. Así primero se ejecuta P1 y cuando finaliza P1 se puede ejecutar P3 y cuando finaliza P3 se puede ejecutar P2 y cuando finaliza P2 se puede ejecutar P1 y así sucesivamente.



4. **Gestión de Impresoras en un Centro de Copiado**

   Imagina un centro de copiado con múltiples impresoras disponibles para los clientes. Los clientes llegan con sus documentos para imprimir y utilizan las impresoras disponibles. Para garantizar un uso eficiente de las impresoras, se implementará un sistema de gestión llamado

"GestorImpresoras". Cada impresora tiene un estado (ocupada o disponible) y se representa como un recurso compartido. Los clientes se modelan como hilos que llegan al centro de copiado y desean imprimir sus documentos.
El GestorImpresoras se encarga de regular el acceso a las impresoras de la siguiente manera:

- Si una impresora está disponible, el cliente puede imprimir su documento de inmediato.
- Si todas las impresoras están ocupadas, el cliente debe esperar hasta que una impresora esté disponible.
- Cada impresión lleva un cierto tiempo, simulado con un retraso.

Diseña un programa en el que:

- Los clientes (hilos) lleguen al centro de copiado y soliciten imprimir un documento.
- Si hay una impresora disponible, el cliente la utiliza y simula el proceso de impresión.
- Si todas las impresoras están ocupadas, el cliente debe esperar hasta que una impresora esté libre.

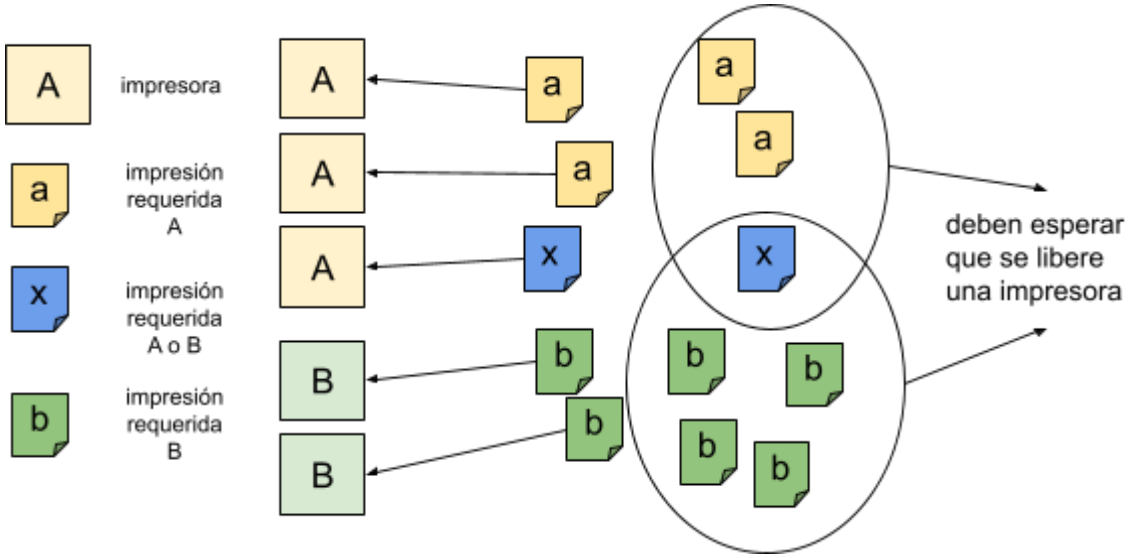
Utiliza semáforos para garantizar el acceso sincronizado a las impresoras.

5. Mejorar el código anterior para que ahora, el centro de impresión cuente con dos tipos distintos de impresoras para dar servicio a los usuarios: impresoras de tipo A e impresoras de tipo B. Obviamente, el número de impresoras de cada tipo es limitado: NumA impresoras de tipo A y NumB impresoras de tipo B. Para imprimir un trabajo en una impresora de un tipo determinado (A o B) es necesario hacer la solicitud indicando el tipo de impresión requerida. A la hora de imprimir los trabajos se pueden considerar tres grupos distintos de procesos usuarios:
   - Los que requieren una impresora de tipo A.
   - Los que requieren una impresora de tipo B.
   - Los que pueden utilizar una impresora de uno cualquiera de los dos tipos.

   Los usuarios generan trabajos y los imprimen. Como restricción: dos hilos no pueden ejecutar simultáneamente las operaciones de impresión (Imprimir A o Imprimir B) sobre una misma impresora.

Cuando un usuario quiera imprimir un trabajo deberá hacerlo sobre una impresora compatible con él y que no esté siendo utilizada por otro usuario. En otro caso el proceso deberá esperar.

Ejemplo: Tenemos 3 impresoras de tipo A y dos impresoras de tipo B. Cada una puede imprimir un trabajo a la vez.



6. Considere los siguientes dos hilos: un taxista y usted.

   Usted necesita un taxi para llegar a su destino y el taxista necesita que un pasajero tome el taxi para cobrar por su trabajo. Por lo tanto cada uno tiene una tarea. Usted espera tomar el taxi y viajar hasta que el taxista le notifique que ha llegado a su destino ya que resultaría molesto para usted y para él preguntar cada dos segundos “¿ya llegamos a destino?”.

   Entre distintos viajes, el taxista quiere dormir en el taxi hasta que otro pasajero necesite ser conducido hasta algún lugar, no desea despertarse de su siesta cada cinco minutos para ver si arribó un pasajero. Por lo tanto, ambos hilos preferirían realizar su trabajo de la manera más relajada posible.

   Usted y el taxi necesitan alguna forma de comunicar sus necesidades al otro. Mientras usted está ocupado caminando por la calle buscando un taxi, el taxista está durmiendo plácidamente en la cabina. Cuando usted le avisa que quiere tomar el taxi, él se despierta y comienza a conducir. Cuando usted arriba a su destino, el taxista le notifica que ha llegado y continúa con su trabajo. El taxista debe ahora esperar y dormir nuevamente una siesta hasta que el próximo pasajero llegue. Diseñe una solución de comunicación de los hilos que

modele dicha situación.

7. **Los Pollos Hermanos**

   Una pequeña empresa de ***k*** empleados dispone de un pequeño lugar donde los empleados pueden tomar algo rápido (por ejemplo 1 café y una porción de pollo frito). La pequeña confitería es atendida por **un mozo** y tiene espacio para **un empleado**.

   Cuando un empleado desea servirse algo se acerca al lugar, si el espacio está libre se queda, e indica al mozo que está allí para que le sirvan. Elige lo que desea de una pequeña lista de opciones, y espera pacientemente que le sirvan. Cuando le han servido, le indican que puede empezar a comer. El empleado come y cuando termina deja la silla libre, agradece al mozo y vuelve a trabajar.
   Por su parte el mozo, cuando no hay empleados que atender, aprovecha a disfrutar de su hobbie de inventar nuevas versiones de pollo, hasta que llegue otro empleado.

   - Escriba un programa que ejecute las funciones descritas utilizando los mecanismos de sincronización que considere necesarios. Debe existir un hilo para el mozo y un hilo para cada empleado. La solución debe estar libre de bloqueos. La solución debe maximizar la concurrencia de los hilos.

8. **Fábrica de producción**

   Imaginemos una fábrica de producción con dos líneas de ensamblaje, una para productos eléctricos y otra para productos mecánicos. Para coordinar el flujo de producción, se implementa un sistema de gestión llamado "**ControladorProducción**".

   Cada línea de ensamblaje tiene su propio equipo de trabajadores que ensamblan los productos. Los productos eléctricos llegan desde el norte, mientras que los productos mecánicos llegan desde el oeste. En la entrada de cada línea hay sensores que detectan la llegada de un producto y también hay sensores que indican cuando un producto sale de la línea de ensamblaje.

   El **ControladorProducción** se encarga de regular el flujo de los productos entre las dos líneas de ensamblaje. Cada producto se representa como un hilo (thread) que invoca los métodos *llegaElectrico()* o *llegaMecanico()* cuando llega al cruce de producción.

   Si la línea de ensamblaje correspondiente está funcionando y tiene capacidad, el producto pasa inmediatamente. Si la línea de ensamblaje está detenida debido a un problema técnico, el producto espera hasta que se reinicie la línea. Cada producto también requiere un cierto tiempo para ser ensamblado. Una vez ensamblado, el producto invoca el método sale() en el **ControladorProducción** para indicar que ha terminado.

   Además, hay un hilo de control que llama al método *cambiaLineas()* cada cierto tiempo para cambiar la configuración de las líneas de ensamblaje, alternando entre la producción de productos eléctricos y mecánicos.

   En resumen, el **ControladorProducción** simula la gestión de la producción en una fábrica con dos líneas de ensamblaje, regulando el flujo de productos entre las líneas y coordinando

el ensamblaje de los mismos.