Relación de ejercicios de Programación Multihilo

1. Crear dos hilos en el que cada uno de ellos escriba una serie de números desde el 0 al 499, también deberá mostrar el identificador del hilo. Cada hilo deberá esperar 500 milisegundos entre número y número escrito.
2. Crear tres hilos en el que cada uno escriba durante su ejecución el siguiente mensaje, donde el color sea introducido por teclado antes de la ejecución del hilo:

   "Hola, el mundo es de color [color]"

3. Crear dos hilos que realicen una cuenta común y consecutiva de números:

   1, 2, 3, 4, 5, …

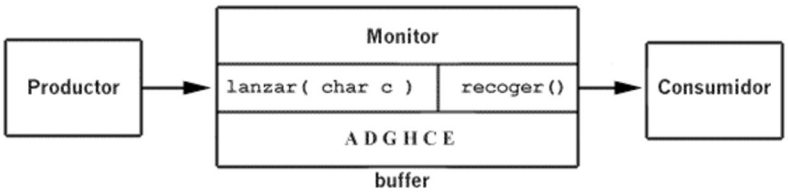
   ¿Qué problema observas en la ejecución?

4. Modifica el ejercicio anterior para que la cuenta se realice de manera alterna. Es decir: Th0-1, Th1-2, Th0-3, Th1-4, …
5. Crear un programa en el que se realice una carrera de cinco camellos en la que tengan que recorrer 25 m. Cada camello corre de manera independiente y al final de la carrera debe mostrarse el podio.
6. Implementa el juego de dados del "Snake-eye" de manera que haya que lanzar dos dados y si se obtiene la puntuación de 2, el jugador ha perdido. En el caso de que el resultado sea de 11 o siete, el jugador ha ganado. En cualquier otro caso, debe lanzar de nuevo los datos.
7. Implementar una solución al problema clásico del Productor-Consumidor, utilizando hilos.

   Nota: Implementa un retardo en la recepción del dato por parte del consumidor.

   ¿Qué ocurre si no insertamos el retardo?

   ¿Qué problemas se nos pueden dar durante la ejecución?



8. Implementar la siguiente situación mediante hilos sincronizados: En un castillo, el rey va a realizar una audiencia a tres caballeros de la corte, Cuando el rey llegue a la audiencia, saludará a sus caballeros diciendo:

"BUENOS DÍAS SÚBDITOS", contestando éstos: "BUENOS DÍAS MAJESTAD". Ningún caballero hablará hasta que no lo haya hecho el rey.

9. Modifica el programa anterior para que el rey no hable hasta que no estén todos los caballeros, y éstos no hablen hasta que el rey no haya saludado.
10. Desarrollar un programa que simule la siguiente situación: Tres amigos (Wellington, Napoleón y Agustina de Aragón), quedan para cenar al estilo de los montes (comer todos del mismo perol con una misma cuchara), por lo que sólo uno puede comer mientras los otros dos esperan.
    Cada uno de ellos comerá una vez en cada ronda, pero no implica que coman siempre en el mismo orden.
11. Implementar mediante un programa Java la simulación de un semáforo en el que la secuencia de luces sea el tradicional Verde – Ámbar – Rojo. La secuencia deberá repetirse 10 veces.
12. Simular mediante un programa java la siguiente situación:

    En una habitación hay una puerta y dos personas que tienen por costumbre abrir y cerrar la puerta constantemente. Pero no podrán abrir la puerta si ya está abierta o cerrarla si ya está cerrada, por lo que deberán esperarse a que ésta esté en la posición apropiada para realizar la acción correspondiente.

    Para ello, se dispondrá de las siguientes clases y métodos:

```
class Habitacion
      abrir(String persona)
      cerrar(String persona)

class Persona

class Puerta //Objeto
```

Ejecutar el programa con el siguiente código principal:

```
      public static void main(String[] args) {
            Habitacion miHabitacion=new Habitacion();
            Persona persona1=new Persona(miHabitacion);
            Persona persona2=new Persona(miHabitacion);
            persona1.setName("Alice");
            persona2.setName("Bob");
            persona1.start();
            persona2.start();
      }
```

Simular los tres posibles casos siguientes:

a) Cada una de las personas abren y cierran.

b) Cada uno abre y cierra aleatoriamente. ¿Qué problema nos podemos encontrar con esta situación?

13. Implementar en Java mediante el uso de hilos sincronizados la siguiente simulación:

    En un paso de peatones los coches deben ceder a los peatones que desean cruzar la calle. De
    esta manera, un peatón no podrá cruzar si un coche está pasando en ese momento. Por su parte,
    un coche no podrá pasar si un peatón está esperando a cruzar o cruzando ya.
    Los coches circularán siempre en orden, no pudiendo pasar por el paso de peatones adelantando
    a otros coches.
    Los peatones sí podrán cruzar antes que otros peatones.
    Implementar la simulación con 10 coches y cinco peatones.

14. Simular mediante un programa java la siguiente situación:

    En una intersección de una carretera, circulan bicis, coches y camiones. Según las señales de tráfico que regulan el cruce, las bicicletas tienen preferencia sobre los coches y camiones, y los coches tienen preferencia sobre los camiones.
    Para ello, se dispondrá de las siguientes clases y métodos:

```
class Cruce
      llegar(Vehiculo miVehiculo)
      cruzar(Vehiculo miVehiculo)


class Vehiculo
```

Ejecutar el programa con el siguiente código principal:

```
      public static void main(String[] args) {
            Cruce miCruce=new Cruce();
            Vehiculo bicicleta1=new Vehiculo("bici1","bici",miCruce);
            Vehiculo coche1=new Vehiculo("coche1","coche",miCruce);
            Vehiculo camion1=new Vehiculo("camion1","camion",miCruce);
```

```
        …
        //hasta cinco vehículos de cada tipo.
}
```

15. Implementa la simulación de movimientos de una cuenta bancaria de forma que, los usuarios podrán ingresar o reintegrar sólo cantidades de 500€. El cajero no permitirá sacar dinero si el saldo actual es inferior a 4000€.
    Simular el ejercicio para dos usuarios (uno para ingresar y otro para reintegrar) y cada uno realizará diez movimientos. El saldo inicial de la cuenta es de 5000 €.
    Utilizar bloqueo de objetos para este ejercicio.
16. Implementar una simulación de un aeropuerto en el que los aviones deban solicitar el despegue, dirigirse a la pista para despegar, realizar el despegue, volar, solicitar el aterrizaje, realizar la maniobra de aproximación y aterrizar.
    El despegue y el aterrizaje deberá realizarse en orden del permiso concedido por la torre de control, no pudiendo aterrizar o despegar un avión si la pista está ocupada o tiene otras aeronaves con el permiso ya concedido.
    Simular la ejecución con 10 aviones.
17. Implementar un programa en Java que simule la siguiente situación:
    Una impresora admite documentos para imprimir de distintos procesos que son gestionados por prioridades, siendo la prioridad 1 la más baja y la 5 la más alta.
    La impresión de un documento depende de la longitud del mismo.
    Cuando un proceso solicita imprimir se pone en cola de impresión según el orden de la prioridad que tenga.
    Un proceso no podrá imprimir mientras que la impresora esté imprimiendo otro documento o no le corresponda porque haya trabajos con más prioridad. En caso de que haya dos procesos con la misma prioridad, el primero que haya solicitado imprimir será el que haga uso de la impresora.
    Notas:
    - Visualizar el momento en el que el proceso entra en el sistema y la prioridad que tiene.
    - Visualizar la cola de impresión siempre que un proceso entre o salga de ella.
    - Visualizar el momento en el que el proceso comienza a imprimir.
    - Visualizar el momento en el que el proceso finaliza de imprimir.
    - Visualizar el momento en el que un proceso intenta colarse en la impresión.

- Simular la entrada de procesos de manera aleatoria entre 1 y 3 segundos.
- Simular el tiempo que tarda en imprimir un proceso de manera aleatoria entre 2 y 4 segundos.
- Simular la ejecución con 20 procesos.

18. Simular mediante un programa Java el funcionamiento de una peluquería en la que el peluquero deberá limpiar la peluquería en los momentos en los que no tenga clientes que atender.

    Los clientes, llegarán a la peluquería y preguntarán si pueden pelarse. Entonces el peluquero les dará el número de orden y serán atendidos según el orden asignado.

    Cuando los clientes sean servidos, darán las gracias y se marcharán de la peluquería.

    Simular la ejecución del programa con 10 clientes, teniendo en cuenta que los clientes llegan a la peluquería con una frecuencia de entre 4 y 5 segundos. Y que el peluquero tarda entre 6 y 7 segundos en pelar a un cliente.

19. Implementar un cronómetro, de manera que muestre el tiempo en el formato MM:SS, como se muestra en la imagen.

```
1:56
1:57
1:58
1:59
2:0
2:1
```

20. Implementa el funcionamiento de un supermercado en el que los clientes entran, realizan su compra (eligiendo los productos de entre los stands de los diferentes pasillos) y al finalizar, se ponen en cola del único cajero que existe. Cuando sea su turno, serán atendidos por el cajero y pagarán, marchándose del local.

    Cada cliente tardará entre 1 y 5 segundos en realizar la compra. Y el cajero tardará entre 1 y 2 segundos en cobrarles los productos adquiridos.

    Simular la ejecución del programa con 20 clientes.

21. Un puente levadizo conecta dos carreteras separadas por un río. Normalmente, los coches cruzan el puente libremente. Sin embargo, cuando un barco se acerca, el puente debe levantarse para permitir el paso del barco, deteniendo temporalmente el tráfico de coches.

    Debes simular este sistema utilizando Java, implementando:

- Hilos que representan coches intentando cruzar el puente.
- Hilos que representan barcos acercándose al puente.
- Sincronización y bloqueo para garantizar que:
  - Mientras el puente esté levantado para que pase un barco, ningún coche pueda cruzarlo.
  - Mientras el puente esté abajo, los coches pueden cruzar libremente, pero no pueden cruzar barcos.
  - Cuando un barco se aproxima, debe avisar por radio al control del puente para que pare el tráfico, debiendo pararse todos los coches que no han entrado en el puente y salir todos los coches que se encuentren cruzando en ese momento.
  - Una vez el barco haya cruzado, el puente debe bajarse, para que el tráfico rodado pueda de nuevo reanudarse.

El ejercicio debe contar con hilos y sincronización. Además de control de errores y buena comunicación con el usuario.

22. En un parque de atracciones, hay una montaña rusa que puede transportar un número limitado de personas por viaje (por ejemplo, 3 personas).

    Los visitantes llegan al parque de manera simultánea (representados por hilos) e intentan subir a la montaña rusa.
    Si el vagón ya está lleno, deben esperar su turno hasta que haya espacio disponible.
    Cuando un visitante logra subir, permanece durante un tiempo en el viaje y luego baja, liberando su lugar para que otro visitante pueda subir.
    Realiza una simulación de 10 personas que quieren subirse a la atracción.

23. En un taller de reparaciones, hay varios mecánicos disponibles para atender a los coches que llegan.

    Cada coche que llega debe esperar a que un mecánico libre lo repare.

    Si hay un mecánico libre, el coche es atendido de inmediato.
    Si todos los mecánicos están ocupados, el coche debe esperar en una cola hasta que uno quede libre.
    Una vez reparado, el coche se marcha y el mecánico queda disponible para el siguiente cliente.

24. En una oficina, varios empleados necesitan usar una única impresora para imprimir sus documentos.

Solo un documento puede ser impreso a la vez.
Si un empleado quiere imprimir pero la impresora ya está en uso, debe esperar hasta que esté libre.
Cuando termina de imprimir, el empleado libera la impresora para el siguiente.

25. Un restaurante tiene una cocina y un área de servicio. En este restaurante, los cocineros preparan los platos y los camareros se encargan de servirlos a los clientes. La interacción entre cocineros y camareros está controlada por una bandeja que tiene una capacidad limitada de platos (por ejemplo, 5 platos). Si la bandeja está llena, los cocineros no pueden colocar más platos hasta que un camarero retire uno.

    El restaurante debe gestionarse de la siguiente manera:

    - Cocineros: Preparan los platos y los colocan en la bandeja. Cada cocinero puede preparar un plato a la vez. Si la bandeja está llena, deben esperar a que haya espacio para colocar un nuevo plato.
    - Camareros: Recogen los platos de la bandeja para servirlos a los clientes. Si la bandeja está vacía, los camareros deben esperar a que haya platos disponibles para recoger.

26. Un edificio tiene varios ascensores que se utilizan para mover a las personas entre los pisos del edificio. Cada ascensor tiene un número limitado de personas que puede transportar a la vez. Además, los usuarios pueden llamar al ascensor desde cualquier piso, y el ascensor debe decidir cuál de los varios ascensores disponibles es el más cercano para atender la llamada.

    Reglas de operación:

    - Ascensores: Los ascensores tienen un límite de capacidad máxima (por ejemplo, 5 personas). Si un ascensor está lleno, no puede aceptar más personas hasta que haya espacio disponible.
    - Usuarios: Los usuarios se encuentran en diferentes pisos del edificio y desean subir o bajar. Cada vez que un usuario necesita ir a otro piso, llama a un ascensor. El sistema debe seleccionar el ascensor más cercano y permitir que el usuario ingrese.
    - Sincronización: Los ascensores y los usuarios deben estar sincronizados de tal manera que:
    - Un ascensor no puede ser llamado simultáneamente por dos usuarios.

- Los ascensores deben estar ocupados sólo por un número limitado de personas (capacidad máxima).
- Los usuarios deben esperar si el ascensor está lleno o si está en otro piso y necesitan esperar a que llegue.
- Notificación y Espera: Cuando un ascensor llega a un piso y está listo para recibir a los usuarios, debe notificarlos y permitirles abordar.