Trabajo práctico Nº 4

Sección crítica

1. Analice el siguiente código:

```
public class DualSynch {
		private Object syncObject = new Object();

		public synchronized void f() {
			for(int i = 0; i < 5; i++) {
				print("f()");
				Thread.yield();
			}
		}

		public void g() {
			synchronized(syncObject) {
				for(int i = 0; i < 5; i++) {
					print("g()");
					Thread.yield();
				}
			}
		}
	}
```

```
public class SyncObject {
		public static void main(String[] args) {
			final DualSynch ds = new DualSynch();

	// solo por cuestiones prácticas se trabaja de esta forma
			Thread hilo = new Thread() {
					public void run() {
						ds.f();
					}
				};
			hilo.start();
			ds.g();
		}
	}
```

a. ¿Cuál es el efecto de Thread.yield()?

b. ¿Cuál es el efecto de "synchronized (syncObject)"?

c. ¿Cuál es el efecto de "synchronized void f()"?

d. ¿Cuál es la diferencia entre el yield() y sleep()?

e. Indique el funcionamiento general de lo presentado.

2. Considere la clase DualSynch modificada, ahora con un recurso compartido por los hilos:

```
public class DualSynch {
	private Object syncObject = new Object();
		int dato=5;

	public synchronized void f() {
		for(int i = 0; i < 5; i++) {
			dato = dato * 4;
			print("f()" + dato);
			Thread.yield();
		}
	}

	public void g() {
	   synchronized(syncObject) {
		for(int i = 0; i < 5; i++) {
			dato = dato + 20;
			print("g()" + dato);
			Thread.yield();
		}
	   }
	}
    }
```

```
public class SyncObject {
	public static void main(String[] args) {
		final DualSynch ds = new DualSynch();
		Thread hilo = new Thread() {
				public void run() {
					ds.f();
				}
			}
		hilo.start();
		ds.g();
	}
    }
```

a. ¿Cuántos hilos hay en ejecución?

b. ¿Cuál es el recurso compartido?

c. la salida, ¿es siempre la esperada?

d. ¿Es posible obtener la siguiente salida?

```
g(), 25
g(), 120
g(), 140
g(), 160
f(), 160
f(), 640
f(), 2560
f(), 10240
f(), 40960
g(), 40980
```

3. Dado el siguiente código:

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)

4. 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);
```

a. ¿Qué sucede si el semáforo sem2 se inicializa en 0?
b. ¿Qué sucede si el semáforo sem1 se inicializa en 1?

5. Implementar el pseudocódigo 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.

6. OBLIGATORIO - Retome el ejercicio 3 del TP 3 (Synchronized), cómo mejoraría el ejercicio para que no haya espera activa o no tenga sleep (xxxx)
    a. *Diseñar un programa en Java para mostrar las letras A, B y C. Para ello, utilizaremos 3 hilos, pero se quiere limitar la posible salida de caracteres por pantalla para que no se produzca de cualquier forma sino con la secuencia predefinidas: ABBCCC, por ejemplo: ABBCCCABBCCC....*
    Se recomienda el uso de una variable compartida por los hilos que sirva para asignar turnos, de manera que cada hilo imprima su letra cuando sea su turno. De esta forma, cuando se crean los hilos, a cada uno se le asigna un identificador para que puedan comprobar cuando es su turno y que mientras no sea su turno, no haga nada.

    b. ***A tener en cuenta:*** *Cada hilo sabe la cantidad de veces que debe imprimir. Por ejemplo: El hiloB sabe que tiene que imprimir 2 veces la letra B.*

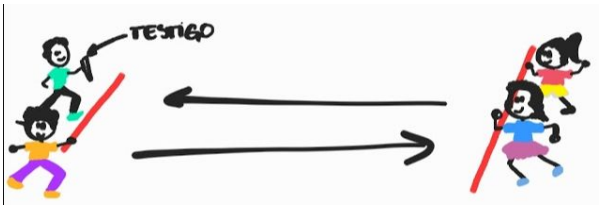
7. En un **centro de impresión** se cuenta con dos tipos distintos de impresoras para dar servicio a los usuarios: impresoras de tipo A e impresoras de tipo B. Obviamente, el número de impresoras de cada tipo es limitado: NumA impresoras de tipo A y NumB impresoras de tipo B. Para imprimir un trabajo 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:
    a. Los que requieren una impresora de tipo A.
    b. Los que requieren una impresora de tipo B.
    c. Los que pueden utilizar una impresora de uno cualquiera de los dos tipos.

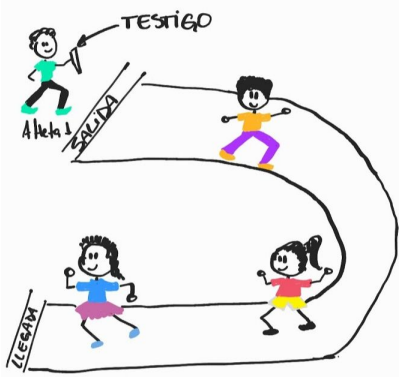
    Los hilos usuarios generan trabajos y los imprimen. Como restricción: dos hilos no pueden ejecutar simultáneamente las operaciones de impresión (Imprimir A o Imprimir B) sobre una misma impresora.
    Cuando un usuario quiera imprimir un trabajo deberá hacerlo sobre una impresora compatible con él y que no esté siendo utilizada por otro usuario. En otro caso el proceso deberá esperar.

8. OBLIGATORIO - Implementar una carrera por relevos:

   Tenemos 4 atletas dispuestos a correr, además se cuenta con una clase principal Carrera y un objeto estático testigo. Todos los atletas empiezan parados, uno comienza a correr (tarda entre 9 y 11s) y al terminar su carrera pasa el testigo a otro que comienza a correr, y así sucesivamente. Diseñe un modelo que represente dicho enunciado. La solución debe estar libre de bloqueos y debe maximizar la concurrencia de los hilos.

   Pistas:

   - Thread.sleep y Math.random para simular la carrera
   - Utilizar un Semaphore como testigo
   - System.currentTimeMillis para ver tiempos
   - Elija una de las siguientes opciones para implementar
     - Opción 1: 

     - Opción 2: 

9. OBLIGATORIO - 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. Resultaría molesto para usted y para el 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 esta 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.

10. Investigue la Interfez *Lock*, y su implementación *ReentrantLock* respecto a su uso para lograr la exclusión mútua. Considere sólo los métodos básicos necesarios.
11. Resuelva el ejercicio 6 del tp2 (Adicionales) utilizando Locks como herramienta de sincronización.
12. OBLIGATORIO - Retome el ejercicio 8 del tp3 (Adicionales) y resuelva utilizando Locks.

    En una tienda de mascotas están teniendo problemas para tener a todos sus hámster felices. Los hámster comparten una jaula en la que hay un plato con comida, una rueda para hacer ejercicio, y una hamaca en la que pueden descansar. Todos los hamsters quieren comer del plato, correr en la rueda y luego descansar en la hamaca. Pero se encuentran con el inconveniente de que solo 1 de ellos puede comer del plato, solo uno puede correr en la rueda y solo 1 puede descansar en la hamaca.

    Implemente un programa para simular la situación planteada, en donde todos los hámster puedan realizar todas las actividades.

    Nota: considere que todas las actividades consumen cierto tiempo, por lo que para la simulación se sugiere asignar ese tiempo con "sleep()".

13. OBLIGATORIO - Una pequeña empresa (Los Pollos Hermanos) de 6 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 1 empleado.

    El mozo se encarga de servir bebidas y el cocinero se encarga de la comida.

    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.

a. 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.

14. OBLIGATORIO - A la empresa “Los Pollos Hermanos” le va muy bien y decide ampliar el espacio de comidas de forma de ofrecer a sus empleados más opciones. Es por ello que agregan un cocinero más, se mantiene el mozo y ahora hay espacio para 2 empleados.

El mozo se encarga de servir bebidas y el cocinero se encarga de la comida. Cuando un empleado desea comer o tomar algo se acerca al comedor. Puede elegir: solo tomar algo, solo comer, o tomar algo y comer. En este último caso debe ser atendido primero por el mozo y recién cuando consigue su bebida puede solicitar la comida.

Cuando se acerca al comedor si existe algún lugar libre se queda, e indica al mozo que está allí para que le sirvan, y espera pacientemente que le sirvan la bebida. Cuando le han servido, le indican que puede continuar y solicitar la comida al cocinero.

El cocinero prepara la comida y le avisa cuando está lista, mientras el empleado espera. El empleado come y cuando termina deja el lugar libre y vuelve a trabajar.

Cuando no hay empleados que atender **el cocinero** aprovecha a ordenar su cocina y probar nuevas recetas, y **el mozo** continúa con su jovi.

a. Modifique lo que sea necesario en la solución del problema anterior para considerar la nueva situación.