

Trabajo práctico N° 3

Sección crítica - Sincronización

- 1) En este ejemplo vamos a tener 3 clases: una clase Main, una clase CuentaBanco y una clase VerificarCuenta (Hilo).
- a) Ejecute el código y comente el resultado. ¿Qué corrección debería realizar para mejorar la protección de los datos?

```
public class Main {
    public static void main(String[] args) {
        VerificarCuenta vc = new VerificarCuenta();
        Thread Luis = new Thread(vc, "Luis");
        Thread Manuel = new Thread(vc, "Manuel");
        Luis.start();
        Manuel.start();
    }
}

public class CuentaBanco {
    private int balance = 50;
    public CuentaBanco(){
    }

    public int getBalance(){
        return balance;
    }

    public void retiroBancario(int retiro){
        balance = balance - retiro;
    }
}

public class VerificarCuenta implements Runnable{
    private CuentaBanco cb = new CuentaBanco();

    private void HacerRetiro(int cantidad)throws InterruptedException{

        if (cb.getBalance() >= cantidad)
            System.out.println ( Thread.currentThread().getName() + "está realizando
un retiro de: " + cantidad + ".");
            Thread.sleep(1000);
            cb.retiroBancario(cantidad);

            System.out.println(Thread.currentThread().getName() + ": Retiro
realizado.");
            System.out.println(Thread.currentThread().getName() + ": Los fondos
```

```
son de: " + cb.getBalance());  
    } else {  
        System.out.println("No hay suficiente dinero en la cuenta para  
realizar el retiro Sr." +  
Thread.currentThread().getName());  
        System.out.println("Su saldo actual es de" +  
            cb.getBalance());  
        Thread.sleep(1000);  
    }  
} // de hacer retiro  
  
public void run() {  
    for (int i = 0; i <= 3; i++) {  
        try {  
            this.HacerRetiro(10);  
            if(cb.getBalance() < 0){  
                System.out.println("La cuenta está sobregirada.");  
            }  
        } catch (InterruptedException ex) {  
            Logger.getLogger(VerificarCuenta.class.getName()).  
                log(Level.SEVERE, null, ex); }  
    }  
}
```

2) Realice un programa en donde se muestre el comportamiento de dos personajes que alteran nuestro puntos de VIDA (recurso compartido) que se inicializa con 10, de la siguiente manera:

El Orco (hilo 1) nos golpea quitándonos (-) 3 de VIDA.

El Curandero (hilo 2) nos cura dándonos (+) 3 de VIDA.

Debe tener en cuenta que las operaciones son: obtener VIDA, operarlo y volverlo a guardar.

- Pruebe la ejecución varias veces e indique el valor resultante en cada una.
- ¿Qué puede concluir?
- Ahora modifique el programa utilizando alguna herramienta para sincronizar la ejecución.

3) Una computadora central conectada a terminales remotas vía links de comunicación se utiliza para realizar reservas de asiento para una sala de concierto.
Para reservar un asiento, un cliente elige un asiento libre y el encargado entra el número del asiento elegido en la terminal, y genera un ticket si está libre.
Se requiere un sistema que evite que se realicen reservas dobles para un mismo sitio, que permita que los clientes elijan libremente entre los asientos libres.



4) 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.

- a) Implemente un programa para simular la situación planteada, en donde todos los hámster puedan realizar todas las actividades.
- b) **Nota:** considere que todas las actividades consumen cierto tiempo, por lo que para la simulación se sugiere asignar ese tiempo con "sleep()".

5) Implemente una clase Auto, subclase de Vehículo con los atributos representativos de auto (patente, modelo, marca, km, etc). Un auto al recorrer varios kilómetros consume combustible, y cuando llega al nivel de la reserva debe cargar combustible para poder continuar. Considere que se desea simular la siguiente situación: Varios autos recorriendo la ciudad y un surtidor que despacha combustible.

. Realice un programa para representar la situación. ¿Cómo representará los autos? ¿y el surtidor? Realice una prueba del programa con 5 autos como mínimo.

- Considere que el surtidor tiene capacidad para *maximo* litros de combustible, que se van gastando a medida que los autos pasan a abastecerse.
- Cada vez que se realiza una carga de combustible escriba un mensaje apropiado para informar el estado actual del surtidor

6) Se dispone de un arreglo de números enteros comprendidos entre 1 y 10 con capacidad para **50.000** números. Se desea disponer de un programa que efectúe la suma de los componentes del arreglo utilizando una estrategia concurrente, que obtenga la suma final mediante sumas parciales calculadas por una serie de hilos independientes.

Escriba un programa que realice la tarea propuesta, y que cumpla con las especificaciones siguientes:

- El arreglo debe llenarse con números enteros entre 1 y 10 generados aleatoriamente. Nota: utilizar la clase **Random** de java.
- Habrá *k* hilos, entre los cuales se dividirá el trabajo a realizar: cada uno debe ocuparse de proporcionar la suma parcial de un segmento del arreglo.
- Habrá un programa principal que creará y rellenará el arreglo de datos, y escribirá el valor final resultante de la suma realizada por los hilos.



7) 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.

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. Podría ocurrir que A tenga que imprimir 4 veces, B 2 veces y C 5 veces, siempre respetando el orden que imprime primero A, luego B y después C (AAAABBCCCCCAAABBCCCCC...)

•