Programación de Procesos y Servicios
UD 1 – Concurrencia
Ejercicios 2

Programación Multihilo y sincronización

3. Crea un programa que trabaje con hilos. El hilo principal deberá lanzar 2 hilos, el primero escribirá por consola 15 veces “Hola “ cada 2 segundos. El segundo hilo escribirá “ mundo!” y el retorno de carro otras 15 veces también cada 2 segundos. Si el hilo principal lanza el segundo con un pequeño retraso (unos 20ms) el texto se mostrará por consola sin percatarnos que lo están escribiendo 2 hilos diferentes. Modifica el programa de manera que el hilo principal interrumpa al primer hilo transcurridos 5s desde el arranque de los dos hilos. La respuesta ante la interrupción debe consistir en la salida y finalización de la ejecución del hilo interrumpido.

4. Crea una aplicación que conste de 2 hilos; el primero el hilo principal de la aplicación Java. El hilo principal deberá lanzar un nuevo hilo encargado de imprimir por consola los siguientes mensajes con un intervalo de 4 segundos entre cada uno de ellos (Mensajes: “Programas”, “Procesos”, “Servicios”, “Hilos”). El hilo principal debe quedar a la espera hasta que termine, mostrando cada segundo que está esperando por la finalización del hilo hijo. La ejecución del programa debe durar 16s ya que son 4 mensajes y 4s de espera por cada uno.
   Para poder reducir la duración de la ejecución el programa debe aceptar por parámetro (método main) el tiempo de espera máximo que el hilo principal esperará a la ejecución del hilo secundario, una vez superado ese tiempo el hilo principal debe interrumpir la ejecución del hilo secundario y a partir de ese momento el hilo secundario mostrará los mensajes restantes sin esperas entre la impresión de los mensajes para finalizar la ejecución del hilo cuanto antes. El hilo principal debe mostrar por consola el mensaje de finalización de la ejecución del programa. Puedes imprimir los mensajes que consideres oportunos para verificar la correcta ejecución del programa. Calcula el tiempo de ejecución del hilo principal y muéstralo por consola. Incluye el nombre del hilo que imprime por consola cada vez que muestres un mensaje de salida.

Ejemplos salida por consola:

```
--Con un tiempo de espera de 3s

Hilo: main. Tiempo de espera: 3s
Hilo: main. Esperando a que el hilo Thread-0 termine
Hilo: main. Todavía esperando...
Hilo: Thread-0. Programas
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Cansado de esperar
Hilo: Thread-0. Procesos
Hilo: Thread-0. Servicios
Hilo: Thread-0. Hilos
Hilo: Thread-0. *** Finalizado ***
Hilo: main. *** Finalizado. Tiempo de ejecución:3s. ***
```

```
--Con un tiempo de espera mayor a 16s, por ejemplo 100

Hilo: main. Tiempo de espera: 100s
Hilo: main. Esperando a que el hilo Thread-0 termine
Hilo: main. Todavía esperando...
Hilo: Thread-0. Programas
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: Thread-0. Procesos
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: Thread-0. Servicios
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: Thread-0. Hilos
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: main. Todavía esperando...
Hilo: Thread-0. *** Finalizado ***
Hilo: main. *** Finalizado. Tiempo de ejecución:16s. ***
```

5. Crea una clase llamada Saldo, con un atributo que indique el saldo disponible. El constructor dará valor inicial al saldo. Crea los métodos get (público) y set (privado) para el saldo incluyendo un sleep() aleatorio. Crea otro método que reciba una cantidad y se la añada al saldo, este método debe informar de quién realiza la operación, la cantidad, el saldo inicial y el final. Este método debe ser definido como synchronized. Crea una clase que implemente Runnable, desde el método run hemos de usar el método que añade la cantidad al saldo. Crea el método main un objeto saldo con un valor inicial. Visualiza el valor inicial. Crea varios hilos que compartan el objeto Saldo, a cada hilo le asignamos un nombre y una cantidad. Lanzamos los hilos y esperamos que finalicen para visualizar el saldo final. Comprueba el funcionamiento de la aplicación si quitando synchronized de la declaración del método.