

## EJERCICIOS PROGRAMACIÓN MULTITHREAD II

1. Implementa una clase *Contador*, con un atributo entero y 3 métodos (incrementa, decrementa, y getValor)

Definir 2 tipos de hilo (hiloInc, hiloDec), uno para decrementar y otro para incrementar dicho contador (ejecutarán un bucle de X iteraciones, en las que se modifique el valor del contador según corresponda, y a continuación un Thread.sleep), que cuenten con un atributo interno de tipo Contador.

En el main, crear 2 hilos, uno de cada clase, que trabajen sobre un mismo objeto Contador.

2. Crea un programa Java que lance 5 hilos, cada uno incrementará una variable contador de tipo entero, compartida por todos, 5000 veces y luego saldrá. Comprobar el resultado final de la variable. ¿Se obtiene el resultado correcto? Ahora sincroniza el acceso a dicha variable.

3. Crea una clase Contador que será el recurso compartido por los hilos (lo que se denomina sección crítica). En esta clase:

- Define un atributo de tipo entero, inicializado a 0
- Un método en el que se devolverá el valor de la variable anterior
- Un método que establece (asigna) el valor actual a la variable

Clase ContadorHilo (el hilo), en donde cada hilo accederá tres veces al recurso compartido Contador y lo incrementará en 1, luego dormirá un tiempo aleatorio en cada iteración.

Clase MainContador (el programa principal, main) que creará 4 hilos (se les pasará el nombre del hilo y un objeto de la clase Contador, es decir, el recurso compartido al que accederá) y se lanzan.

4. Crea una clase *Saldo*, con un atributo que indique el saldo, y el constructor en el que se dará un valor inicial al saldo. Contendrá también varios métodos:

- Uno para obtener el saldo (incluir sleep)
- Otro para modificarlo (incluir sleep)
- Otro que realice un ingreso. Recibe una cantidad y se la añade, informando por pantalla de quién ha realizado ese ingreso y la cantidad resultante después del ingreso.

Crear otra clase Thread, que realice ingresos de saldo desde el run.

En el *main*, crear un objeto compartido Saldo por todos los hilos. Crear 3 hilos, cada uno con un nombre. Esperar a la finalización de todos los hilos para mostrar el valor final del saldo.

5. Crear una clase Cuenta, con un atributo saldo y 3 métodos:

- Uno que devuelva el importe del saldo
- Otro que reste al saldo el importe a retirar de la cuenta (modifica el saldo)
- Otro que realice las comprobaciones para verificar que se puede efectuar la retirada (el importe final debe ser 0 o superior), enviando en caso contrario el aviso correspondiente.

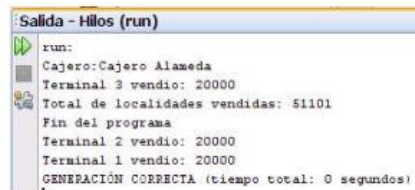
Posteriormente, crear otra clase denominada RetiradaDeCajero, que extienda de Thread. Cada hilo que se cree de esta clase, recibirá un nombre concreto al ser creado, y trabajará sobre un objeto Cuenta. Dentro de su método run(), se intentará realizar un número X de retiradas.

En el main, crear 2 hilos, uno para cada titular de la cuenta (Pepe, María).

*Observaciones: el objeto Cuenta, será compartido por varios hilos.*

6. Diseña un programa que simule la gestión de un punto de venta con tres terminales que acceden a un objeto Cajero que mantiene la cuenta de las localidades vendidas.
- Clase Cajero: es el recurso compartido por los terminales (sección crítica). Contendrá un atributo localidades que llevará la cuenta de las localidades vendidas, que irán incrementando los diferentes terminales. Además, contará con un método mostrarLocalidades. En el constructor simplemente se inicializa el cajero dándole un nombre.
  - Clase Terminal: Su funcionalidad radica en ir vendiendo localidades (hasta un máximo de 20000), anotar el cambio en el contador general de localidades vendidas. Además, cada terminal mostrará por pantalla el total de localidades vendidas por él.
  - MainCajero: programa principal desde el que creamos un cajero y 3 terminales que trabajen con ese cajero. Se arrancan los terminales y cuando terminen mostrarnos un mensaje que nos indique cuántas localidades han vendido en total(debería mostrar 60000, en ejecución sincronizada).

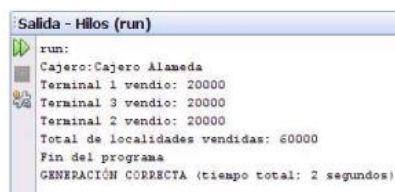
Una posible salida sin sincronización es:



```
run:
Cajero:Cajero Alameda
Terminal 3 vendio: 20000
Total de localidades vendidas: 51101
Fin del programa
Terminal 2 vendio: 20000
Terminal 1 vendio: 20000
GENERACIÓN CORRECTA (tiempo total: 0 segundos)
```

Como puedes ver los datos son incorrectos, debería mostrar que el total de localidades vendidas fue de 60000, pero como no hay un acceso sincronizado al cajero se pierden sumas.

Intenta hacerlo bien, es decir, empleando sincronización, para obtener una salida similar a:



```
run:
Cajero:Cajero Alameda
Terminal 1 vendio: 20000
Terminal 3 vendio: 20000
Terminal 2 vendio: 20000
Total de localidades vendidas: 60000
Fin del programa
GENERACIÓN CORRECTA (tiempo total: 2 segundos)
```

7. Se trata de simular el juego para adivinar un número. Se crearán varios hilos, que actuarán como los jugadores que tienen que adivinar el número. Habrá un árbitro que generará el número a adivinar, comprobará la jugada del jugador y averiguará a qué jugador le toca jugar. El número tiene que estar comprendido entre 1 y 10.

Por lo tanto, existirán 3 clases:

- Árbitro: Contiene el número a adivinar, el turno y muestra el resultado. Se definen los siguientes atributos: número total de jugadores, turno, número a adivinar y si el juego acabó o no. En el constructor se recibe el número de jugadores, y se inicializan el número y el turno. Tendrá métodos para consultar el turno, si el juego ha acabado o no y otro que compruebe la jugada del jugador y averigüe a qué jugador le toca a continuación. Este último método recibirá el ID del jugador y el número que ha jugado. También se indicará si el juego ha finalizado porque el jugador ha acertado el número.
- Jugador: Su constructor recibe un ID de jugador y el árbitro. Lo que harán los jugadores será comprobar si es su turno, en ese caso, generará un número entre 1 y 10 y creará la jugada. Este proceso se repetirá hasta que el juego acabe.
- Principal: Inicializa el árbitro y lanza los hilos.

Un ejemplo de salida sería el siguiente:

NÚMERO A ADIVINAR: 3  
Jugador1 dice: 9  
    Le toca a Jug2  
Jugador2 dice: 9  
    Le toca a Jug3  
Jugador3 dice: 10  
    Le toca a Jug1  
Jugador1 dice: 4  
    Le toca a Jug2  
Jugador2 dice: 7  
    Le toca a Jug3  
Jugador3 dice: 7  
    Le toca a Jug1  
Jugador1 dice: 6  
    Le toca a Jug2  
Jugador2 dice: 3  
    Jugador 2 gana, adivinó el número!!!