# PRÁCTICAS – TEMA 2

## Ejercicios

1. Crea un programa que:
   1. Genere 10 hilos.
   2. Los hilos han de almacenar un nombre que será, por ejemplo “Hilo 1” correspondiendo el número de hilo con la posición en la que se ha creado en el programa.
   3. Cada hilo debe de imprimir un número de veces que se le indique mediante parámetro el mensaje: “Hola soy el hilo … con id … , prioridad … y en mi grupo hay … hilos”
   4. Se han de utilizar los métodos propios de la clase Thread.
2. Crea un programa que tenga las siguientes características:
   1. Una clase de las que tiene es un hilo y tiene un atributo llamado tipo.
   2. Si cuando se crea el hilo se le pasa el tipo 1 muestra los números del 1 al 30.
   3. Si cuando se crea el hilo se le pasa el tipo 2 muestra las letras de la “a” a la “z”.
   4. El main creará un hilo de cada tipo y los ejecutará.
   5. Ejecuta el programa varias veces. ¿El orden que se obtiene es el mismo? ¿Por qué?
3. Modifica el ejemplo 3 y el ejercicio 1 para que al finalizar todos los hilos el main imprima una vez “Los hilos han finalizado”.
4. Haz un programa con las siguientes características:
   1. Los hilos que se creen han de imprimir su nombre acompañado de un mensaje un número de veces especificado. Cuando solo queden 2 hilos vivos (uno y el del main) ha de cambiar el mensaje y decir ¡Soy el último superviviente!
   2. Se deben ejecutar, al menos, 5. Y entre mensaje y mensaje el hilo espera un número aleatorio de segundos.
   3. Al finalizar la ejecución de los hilos el main ha de mostrar el mensaje ¡programa finalizado!
5. Crea un programa que simule el funcionamiento de las cajas de un supermercado:
   1. El supermercado tendrá 10 cajeros.
   2. En el supermercado hay varios clientes, cada uno ha comprado varios productos y hacen cola para que el cajero les cobre.
   3. El cajero ha de atender a los clientes en orden y de manera atómica (primero al cliente 1, luego al cliente 2, etc.).
   4. Cada cliente tendrá un número aleatorio de productos (Entre 1 y 20)
   5. El cajero tardará en pasar por caja cada producto un número aleatorio entre 0.1 y 0.3 segundos (usar la función sleep, es importante tener en cuenta que la función sleep en java pide como parámetros los milisegundos a dormir y no los segundos).
   6. Cada cajero tendrá que atender un número aleatorio de clientes (entre 5 y 10).
   7. Antes de atender un cliente el cajero le saluda, dice su nombre, dice el número de clientes que ha atendido hasta ese momento y el número de segundos que lleva trabajando.
   8. Después de atender un cliente el cajero le despide, dice su nombre, el número de clientes que ha atendido hasta ese momento y el número de segundos que lleva trabajando.
   9. Al finalizar de trabajar el cajero ha de decir su nombre, el número de clientes que ha atendido y el tiempo que ha trabajado.
   10. Al finalizar de trabajar todos los cajeros, el main ha de decir el tiempo que han estado trabajando.
   11. Ejecuta el programa varias veces. ¿Cuánto tarda en ejecutarse el programa?¿A qué se ajusta?





Texto

Descripción generada automáticamente

Fragmento de ejecución del programa

Texto

Descripción generada automáticamente

1. En el ejemplo 5 haz que el hilo espere 1 segundo después de imprimir que está en el hilo. ¿Qué pasa?
2. Vamos a crear una ruleta rusa de hilos:
   1. El main crea diez hilos.
   2. Los hilos imprimen su número de hilo y duermen un segundo.
   3. El main cada segundo ha de interrumpir un proceso de manera aleatoria.
   4. Si el main elimina un proceso que ya está muerto generará otro número aleatorio hasta que el número coincida con uno de los que están vivos.
   5. El hilo, cuando sea el último superviviente, ha de indicar: “soy el hilo x y soy el último superviviente”.
   6. El main, una vez mate todos los hilos, indica que todos los hilos han muerto.
3. Haz un programa con las siguientes características:
   1. Habrá 3 clases (Contador, HiloSuma e HiloResta) más el main.
   2. El main tendrá una variable de la Clase contador con valor 20000.
   3. La clase contador tendrá un valor y 3 métodos. Un getter, uno que incrementa en uno su valor y uno que decrementa en uno su valor.
   4. El HiloSuma ha de llamar al método suma del contador 10000 veces.
   5. El HiloResta ha de llamar al método resta del contador 10000 veces.
   6. El main ha de mostrar el valor final del contador.
   7. ¿Qué ha pasado?
4. Haz un programa con las siguientes características:
   1. Habrá 2 clases (Cuenta e HiloSacarDinero) más el main.
   2. Cuenta tiene una variable con una cantidad de dinero y un método llamado retirarDinero.
   3. retirarDinero solicita un nombre y un dinero a retirar, comprueba que la cuenta no se vaya a quedar en negativo e imprime por pantalla la persona que ha retirado el dinero el antiguo saldo y el saldo actual.
   4. Si la cuenta fuera a quedar en negativo imprime que no se puede retirar la cantidad indicada y el saldo actual.
   5. La cuenta tendrá inicialmente 40 euros.
   6. Los hilos intentarán sacar 4 veces 10 euros.
   7. Habrá 2 hilos.
   8. Para acentuar el problema de sincronización, añade los siguientes sleeps:
      1. Uno aleatorio (hasta 1 segundo) dentro del método retirarDinero (tiene que hacerse una vez comprobada la cantidad).

Texto

Descripción generada automáticamente

* + 1. Uno antes de que cada hilo saque dinero:

Texto

Descripción generada automáticamente

• NO SINCRONICES EL PROGRAMA

¿Qué problemas tienes?

1. Reformula el ejercicio de los cajeros, de la siguiente manera:
   1. En vez de haber una cola por cada cajero hay una única cola con 100.
   2. Los cajeros, según hayan terminado de atender a un cliente, llamarán al siguiente.
   3. Haz que al final impriman la cantidad de clientes atendidos para comprobar que la suma total de todos los clientes atendidos sea 100 y, por tanto, que el programa está sincronizado correctamente.
   4. IMPORTANTE NO SOBRESINCRONIZAR
2. Modifica el ejercicio del PING-PONG para que la salida por pantalla se muestre correctamente.
3. Usando el modelo productor-consumidor crea un productor que lea los caracteres de un fichero de texto cuyo nombre se pase al constructor.
   1. El consumidor ha de obtener los datos que se producen y mostrarlos por pantalla. Por tanto, el mensaje que se muestre por pantalla ha de ser el mismo que el leído del texto.
   2. Al acabar la ejecución ha de indicar que se ha finalizado el programa.
4. Adapta el ejercicio de los cajeros de tal manera que:
   1. Los clientes los genera un productor y que tarda 0,1 s en generar cada uno.
   2. Los cajeros son consumidores de estos clientes.
5. El alumno ha de realizar un juego cuyo fin es adivinar un número. Un árbitro ha de crear un número aleatrorio (entre 1 y 50) y los jugadores intentarán adivinarlo.
   1. Jugador:
      1. Cada jugador es un hilo.
      2. Hasta que el juego finalice estará a la espera de que sea su turno.
      3. Un turno constará de una jugada de cada uno de los jugadores.
      4. Los jugadores tendrán que realizar su jugada en orden.
      5. Una jugada consistirá en el cálculo de un número aleatorio entre 1 y 50.
      6. Una vez el jugador realice la jugada el árbitro la consumirá y comprobará si ha acertado.(Daos cuenta de que esto es un modelo productor-consumidor de manual)
      7. Si no le toca realizar jugada esperará a que el jugador correspondiente realice su jugada y después volverá a comprobar si le toca jugar.
      8. Si otro jugador ha ganado lo notifica y finaliza su ejecución.
      9. Si ha ganado el jugador lo notifica y finaliza su ejecución.
   2. Árbitro:
      1. El árbitro es un hilo.
      2. En cada turno espera hasta que el jugador correspondiente realiza un intento.
      3. Una vez se ha realizado este intento, comprueba si el resultado obtenido del jugador es correcto.
      4. Si es correcto indica que el jugador ha ganado y finaliza el juego.
      5. Si no es correcto llama al siguiente jugador para que realice su jugada.
      6. Cada vez que acaba un turno lo indica “TURNO 1”, “TURNO 2”, etc.
      7. El juego no comienza hasta que todos los jugadores y el árbitro estén listos.
      8. Tanto los jugadores como el árbitro han de finalizar.