PRÁCTICA 2 - PROGRAMACIÓN MULTITHREADING - Primera Parte

Coordinación de Hilos (sincronizar, `wait`/`notify`/`notifyAll`)

Escribe en Java los programas necesarios:

- Utiliza de manera adecuada la Programación Orientada a Objetos.
- Presta atención a las excepciones que pueda originar el código e informa al usuario.
- Imprime en pantalla los mensajes necesarios cuando haya que informar al usuario.
- No olvides comentar el código y documentar los métodos.

1. EJERCICIO 1: CÁLCULOS BÁSICOS

DesarrollA una nueva clase `Calculate` con un método `newValue` y todos los campos necesarios para que funcione de la siguiente manera:

- `newValue(n)`, será llamado por una serie de hilos ejecutores. Este método obtiene un valor entero como argumento y agrega este valor a la variable compartida `totalSumValue`. También calcula el promedio y almacena su valor en la variable averageValue.

De la clase principal:

- Pregunta al usuario cuántos hilos se van a ejecutar (debe ser un número entre 50 y 100) y el valor que se va a enviar al método `newValue(...)`. Este valor será siempre el mismo para todos los hilos.
- Crea una instancia de la clase `Calculate`.
- Crea e inicia el número necesario de hilos.
- Espera a que los hilos terminen.
- Imprime la suma total de valores y el promedio.

> Nota: Comprueba **que siempre** se obtienen los resultados esperados.

2. EJERCICIO 2: COLA DE MENSAJES

Escribe un programa para administrar una cola de mensajes *threadsafe*:

- Crea un objeto de tipo cola de mensajes (Esta clase tendrás que programarla) donde se pueda poner un mensaje y obtener un mensaje en modo *threadsafe* (el mensaje es de tipo *String*). Utilice un objeto `ArrayList` para almacenar los mensajes. Analiza si los métodos `put` and `get` deben de ser bloqueantes o no (Debemos de usar wait/notify o no).
- Para hacer las cosas sencillas, define las siguientes constantes en la clase principal:
  - QUEUE_SIZE: límite superior de mensajes que puede contener la cola de mensajes
  - MAX_WRITTEN_MESSAGES: número de mensajes a escribir en total.
  - NUM_READERS: Número de hilos de lector
  - NUM_WRITERS: Número de hilos de escritor

- Crea e inicia 5 hilos que pongan mensajes en la cola (`put`) a intervalos aleatorios (el mensaje consiste en el nombre del hilo y un número de secuencia, para que se pueda revisar).
- Crea e inicia 3 hilos que **lean** (`get`) e impriman mensajes de la cola. Una vez que se lea un mensaje, se elimina inmediatamente. Cuando el lector termina, imprime el número de mensajes leídos.

> Nota: Se valorará positivamente (como extra) la clase cola de mensajes desarrollada implemente la interfaz java.util.Queue y que al utilizarla en el programa principal, se utilice la interfaz para trabajar con ella en lugar de utilizar la clase.

3. EJERCICIO 3: LA CARRERA DEL TESTIGO LOCO (OPCIONAL)

Se ha de desarrollar un programa en el cual hay 10 corredores y un único testigo. El juez lanzará un testigo al aire y el primer corredor que lo coja correrá, luego, el siguiente corredor cogerá el testigo. La carrera finaliza cuando todos los corredores han corrido por lo menos una vez.

- Cree e inicie 10 hilos de corredor. Para correr, debe coger el testigo.
- El programa inicia el juez que sostiene el testigo durante 5 segundos. Mientras tanto, todos los hilos del corredor esperan. Cuando el tiempo expira, el juez lanza el testigo al aire y uno de los hilos que se encuentra a la espera (aleatorio) coge el testigo y comienza su carrera.
- Cuando un corredor toma testigo, corre aleatoriamente entre 2 y 5 segundos, luego lanza el testigo al aire (otro de los hilos de espera atrapa el testigo y comienza a correr) y termina.
- El juez espera hasta todos los hilos de corredor haya terminado y luego informa de la orden de llegada.