Actividad 2: Threads

Ejercicio 1: Concurrencia

- Vamos a crear un pequeño programa que haga lo siguiente:
    - Crea una clase “Trabajador” que tenga un atributo String “nombre” y método “realizarTarea”. Dicho método simplemente recorrerá un bucle 6 veces y en cada vuelta del bucle imprimirá por pantalla “Estoy en el paso X”, donde X será el valor del índice del bucle donde nos encontremos. Al terminar el bucle escribirá en pantalla: nombre del trabajador + ” he terminado de trabajar”. A continuación crea en el método main un programa con dos instancias de trabajadores, dales un nombre diferente, y llama a sus métodos “realizarTarea”. ¿Qué sale en la consola?
    - Ahora crea otra clase “TrabajadorHilo” en otro archivo que herede de la clase Thread (extends Thread). Va a ser igual que la clase anterior, pero ahora tienes que cambiar algunas líneas del método “realizarTarea”:
        - Lo primero es que la funcionalidad para que el hilo sepa que debe ejecutarla debe ir dentro de un método “ public void run()”; por lo que renombra el método “realizarTarea” de esta clase.
        - En el bucle, vamos a meter un retardo en la tarea. Para ello, además de escribir lo mismo que en la clase anterior, tras escribir en pantalla vamos a hacer que dicho método espere un número aleatorio de milisegundos. Busca documentación sobre el método “sleep” de la clase Thread o algún ejemplo que pueda ayudarte.
        - Tras esto, en la clase main crea las nuevas instancias dándoles nombres diferentes y después tendrás que llamar para cada una al método “start” que es el método que hace que el hilo se ponga en funcionamiento. ¿Qué sale en la consola con esta nueva versión?

Ejercicio 2: Sincronización

- Vamos a hacer un programa donde dos hilos imprimen números en orden del 1 al 10; uno de los hilos imprime números pares, mientras que el otro imprime números impares.
    - Creamos una clase ImpresoraNumeros que tiene un método imprimirNumero() que imprime números en el orden descrito. Esta clase tendrá como atributo el valor numérico que va a ser impreso justo en ese momento. Los hilos pertenecen a la clase ImpresionNumeros que implementará la interfaz Runnable (implements Runnable), y la cual tendrá como atributo una instancia de ImpresoraNumeros, un booleano que marque qué hilo es (hilo 1 = true; hilo 2 = false, por ejemplo) y llamará al método

imprimirNumero() desde el método run de la interfaz. ¿Qué resultados estás viendo en la pantalla?

- Ahora prueba a hacer estos cambios:
  - El método imprimirNumero() márcalo como “synchronized”. Con esto vamos a conseguir que sólo un hilo pueda estar al mismo tiempo ejecutando este método.
  - Ahora tenemos que hacer que los hilos se esperen el uno al otro; es decir, cuando el hilo 1 haya impreso en pantalla el valor impar que le toque, tendrá que liberar el método y esperar a que el hilo 2 imprima y haga lo mismo; así hasta que terminen. Revisa los métodos notify() y wait() que seguro que te ayuda. ¿Cómo se ven ahora los resultados?

Ejercicio 3: Carreras de coches

- Crea un programa Java donde varios hilos compitan entre sí. Tendrás que simular una carrera de coches donde cada hilo representa un coche (extends Thread) y además contendrá el nombre de dicho vehículo. Asegúrate de que los hilos compitan de manera justa y sincronizada. Cada hilo acumulará una distancia recorrida, incrementada de forma aleatoria entre 50 y 100. Cada 400 de distancia acumulada dicho coche descansará durante un tiempo aleatorio entre 1000 y 1500 milisegundos, y se notificará por consola el descanso del coche junto con su nombre y la distancia acumulada hasta el momento. Los coches van llegando a la meta al alcanzar los 1000 recorridos e imprimirán en consola su llegada a meta para saber en qué orden han ido llegando.