Trabajando con Hilos

Ejercicio 4

Crea un programa que contenga dos clases, **Principal** y **Escritora**. La clase **Principal** contiene el main, que instanciará dos hilos de tipo **Escritora** y los ejecutará. Cada hilo hará lo siguiente (indefinidamente):

- Si se inicializa con un True, escribirá números del 1 al 30.
- Si se inicializa con un False, escribirá letras de la 'a' a la 'z'

Comprueba que las salidas de ambos hilos salen mezcladas por la consola.

Solución: Ejercicio4.zip

Ejercicio 5

Crea un programa que contenga dos clases, **Principal** y **DetonadorConRetardo**. La clase **Principal** contiene el main, que instanciará cuatro hilos de tipo **DetonadorConRetardo** y los ejecutará. Cada hilo hará lo siguiente:

- Se le inicializa con un nombre y un valor numérico (contador).
- Cuando el hilo se ejecute, escribe su nombre y el contador. A continuación, reduce el valor del contador en 1. Repite estas acciones hasta que el valor de contador sea 0
- El completar su tarea, informa de que ha finalizado.

El hilo principal del programa (main) **NO** debe de finalizar antes que los demás hilos.

Solución: Ejercicio5.zip

Ejercicio 6

Tomando como base las clases **FrameCronometro** y **RelojCronometro**, crea un programa que funciona como un cronómetro. Cuando se pulse el botón, se lanzará el hilo **RelojCronometro**. Esta clase en su método run () tendrá un bucle infinito que irá mostrando las horas, minutos y segundos que van pasando. Obviamente, será necesario hacer un retardo de un segundo antes de actualizar el cronómetro.

Para comunicar el frame y el hilo se hace uso del **Patrón Observer**. Sin entrar en detalles, este patrón hace que la variable labelCronometro de **FrameCronometro** se actualice ella sola por medio del método update() cuando se le actualiza desde **RelojCronometro**. Para que funcione:

- En cuanto instancias **RelojCronometro** debes registrarlo mediante la siguiente instrucción:

  relojCronometro.addObserver(this).

- Cada vez que desees notificar desde **RelojCronometro** que ha habido cambios y se debe actualizar labelCronometro, se debe indicar así:

  ```
  this.setChanged();
  this.notifyObservers(tiempo);
  this.clearChanged();
  ```

Solución: Ejercicio6.zip

Ejercicio 7

Siguiendo la lógica del ejercicio anterior, las clases **FrameCarreraTortugas** y **Tortuga**, crea un programa que permita hacer carreras de tortugas. Al presionarse el botón, se generarán cuatro hilos Tortuga que, cada segundo, avanzarán un % aleatorio entre 1 y 15.

Para realizar correctamente el ejercicio, fíjate en **FrameCarreraTortugas** los hilos se guardan en un Array de Threads. Los métodos a completar en esta clase son Update, botonIniciarActionPerformed y finalizar.

Solución: Ejercicio7.zip