2.5. GESTION DE HILOS

En ejemplos anteriores hemos visto como crear y utilizar los hilos, vamos a dedicar un apartado a los pasos vistos anteriormente.

2.5.1. CREAR Y ARRANCAR HILOS

Para crear un hilo extendemos la clase **Thread** o implementamos la interfaz **Runnable**. La siguiente línea de código crea un hilo donde *MiHilo* es una subclase de Thread (o una clase que implementa la interfaz **Runnable**), se le pasan dos argumentos que se deben definir en el constructor de la clase y se utilizan, por ejemplo, para iniciar variables del hilo:

```
MiHilo h = new MiHilo("Hilo 1", 200);
```

Si todo va bien en la creación del hilo tendremos en h el objeto hilo. Para arrancar el hilo usamos el método **start()** de esta manera si extiende **Thread**:

```
h.start();
```

Y si implementa **Runnable** lo arrancamos así:

```
new Thread(h).start();
```

Lo que hace este método es llamar al método **run()** del hilo que es donde se colocan las acciones que queremos que haga el hilo, cuando finalice el método finalizará también el hilo.

2.5.2. SUSPENSION DE UN HILO

En ejemplos anteriores usamos el método **sleep()** para detener un hilo un número de milisegundos. Realmente el hilo no se detiene, sino que se queda "dormido" el número de milisegundos que indiquemos. Lo utilizábamos en los ejercicios de los contadores para que nos diese tiempo a ver cómo se van incrementando de 1 en 1.

El método suspend() permite detener la actividad del hilo durante un intervalo de tiempo indeterminado. Este método es útil cuando se realizan applets con animaciones y en algún momento se decide parar la animación para luego continuar cuando lo decida el usuario. Para volver a activar el hilo se necesita invocar al método resume().

El método suspend() es un método obsoleto y tiende a no utilizarse porque puede producir situaciones de interbloqueos. Por ejemplo, si un hilo está bloqueando un recurso y este hilo se suspende puede dar Lugar a que otros hilos que esperaban el recurso queden "congelados" ya que el hilo suspendido mantiene los recursos bloqueados. Igualmente el método resume() también está en desuso.

Para suspender de forma segura el hilo se debe introducir en el hilo una variable, por ejemplo suspender y comprobar su valor dentro del método run(), es lo que se hace en la llamada al método suspender.esperandoParaReanudar() del ejemplo siguiente. El método Suspende() del hilo da valor true a la variable para suspender el hilo. El metodo Reanuda(da el valor false para que detenga la suspensión y continue ejecutándose el hilo:

```
class MyHilo extends Thread {
private SolicitaSuspender suspender = new SolicitaSuspender();
                    //petition de SUSPENDER HILO
public void Suspende() {supender.set(true); }
                    //petición de CONTINUAR
public void Reanuda(){ suspender.set(false); }
```

```
public void run()  {
        try{
                while(haya trabajo por hater)        {
                .....
                suspender.esperandoParaReanudar(); //comprobar
        }
        catch    (InterruptedException exception)        {}
}
}//
```

Para mayor claridad, se envuelve la variable (a la que se hacía alusión anteriormente) en la clase SolicitaSuspender, en esta clase se define el método set() que da el valor true o false a la variable y llama al método notifyAll(), este notifica a todos los hilos que esperan (han ejecutado un wait()) un cambio de estado sobre el objeto. En el método esperandoParaReanudar(se hace un wait() cuando el valor de la variable es true, el método wait() hace que el hilo espere hasta que le llegue un notify() o un notifyAll();

```
public class SolicitaSuspender{
        private boolean suspender;

        public synchronized void set(boolean b)        {
        suspender= b;       //cambio de estado sobre el objeto
        notifyAll();
}
public synchronized void esperandoParaReanudar()
                throws InterruptedException{
        while(suspender)
                wait();    //SUSPENDER HILO HASTA RECIBIR notify()  o  notifiA11()
}
}//
```

El método wait() solo puede ser llamado desde dentro de un método sincronizado (synchronized). Estos tres métodos: wait(), notify() y notifyAll(), se usan en sincronización de hilos. Forman parte de la clase Object, y no parte de Thread como es el caso de sleep(), suspend() y resume(). Se tratarán más adelante.

ACTIVIDAD 2.4

Partimos de las clases anteriores MyHilo y SolicitaSuspender. Vamos a modificar la clase MyHilo. Se define una variable contador y se inicia con valor 0. En el método run() y dentro de un bucle que controle el fin del hilo mediante una variable, se incrementa en 1 el valor del contador y se visualiza su valor, se incluye también un sleep() para que podamos ver los números. Haz una llamada al método esperandoParaReanudar() para suspender el hilo. Crea en la clase un método que devuelva el valor del contador. Al finalizar el bucle visualiza un mensaje.

Para probar las clases crea un método main(), en el que introducirás una cadena por teclado en un proceso repetitivo hasta introducir un *. Si la cadena introducida es S se suspender el hilo, si la cadena es R se reanudará el hilo. El hilo se lanzará solo una vez después de introducir la primera cadena. Al finalizar el proceso repetitivo visualizar el valor del contador y finalizar el hilo. Comprueba que todos los mensajes se visualicen correctamente.