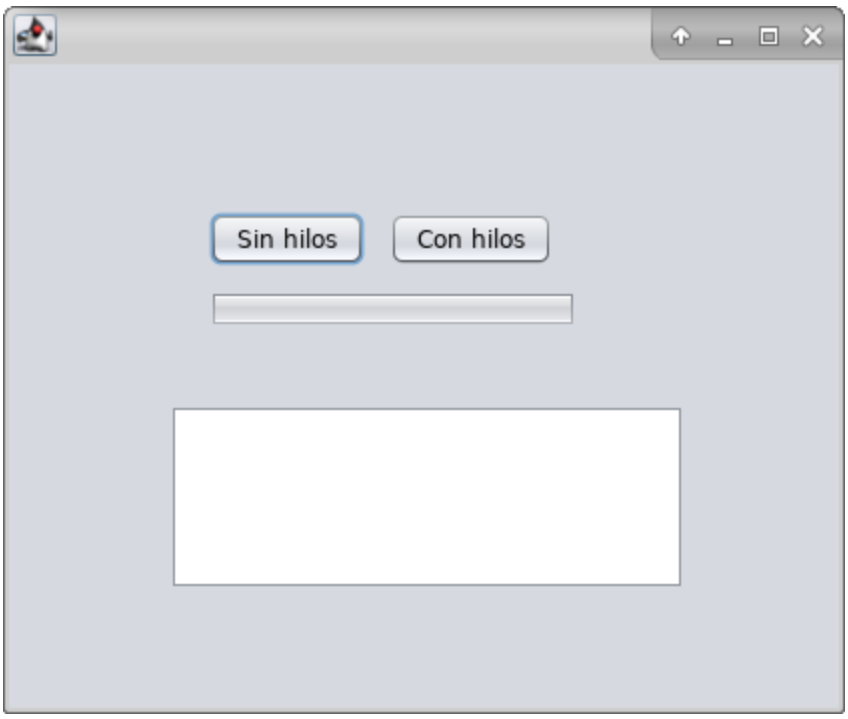
Introducción

Cuando creamos una aplicación de escritorio con Swing utilizaremos botones, cuadros de texto, etc… Sin embargo, **toda aplicación Swing utiliza hilos internamente**. Esto significa que si necesitamos resolver una tarea larga **podríamos dejar «bloqueada» la ventana principal mientras resolvemos la tarea**. Esto produce una sensación muy pobre en el usuario y además puede impulsarle a cerrar la aplicación creyendo que *«se ha colgado»* . Para evitarlo podemos crear aplicaciones Swing que utilicen hilos de manera que el interfaz principal se mantenga activo aunque se esté ejecutando una tarea larga en segundo plano. A modo de ejemplo, vamos a crear una aplicación que sume todos los números de un intervalo dado y que tenga un interfaz como el siguiente:



Paso 1: el interfaz

Empieza por crear un proyecto nuevo y diseña una interfaz como la anterior. En este interfaz tenemos los siguientes controles:

- El boton `btnSinHilos`.
- El boton `btnConHilos`.
- Una barra de progreso que llamaremos `pbProgreso`.
- Un área de texto que llamaremos `txtAreaTexto`.

Paso 2: el bloque del interfaz

Vamos a simular que al pulsar el boton btnSinHilos se ejecuta una tarea larga. Para ello vamos a asociarle a dicho botón este código de gestion de eventos:

```
private void btnSinHilosMouseClicked(java.awt.event.MouseEvent evt) {
   int suma=0;
   for (int i=0; i<2000; i++){
      suma=suma+i;
      try {
         Thread.sleep(1);
      } catch (InterruptedException ex) {
         System.out.println("Error, hilo interrumpido");
      } /*Fin del catch*/
   } /* Fin del for*/
}
```

Si ejecutamos nuestra aplicación,despues pulsamos el botón btnSinHilos e intentamos escribir en el cuadro de texto veremos que **NO OCURRE NADA** . El programa no está mostrando ningún progreso aunque en realidad si «toma nota de las teclas pulsadas» y nos las mostrará tan pronto acabe la suma. Sin embargo es evidente que este comportamiento no es muy recomendable.

Paso 3: la clase SwingWorker

La clase `SwingWorker<Tipo1, Tipo2>` es una clase genérica diseñada para heredar de ella y ejecutar tareas en segundo plano. En ella

- El `Tipo1` es el tipo del resultado que devolveremos.
- El `Tipo2` es el tipo que se utilizará para medir el progreso.

Supongamos que deseamos hacer la misma suma, ir mostrando el progreso y al terminar devolver un mensaje con el resultado en forma de cadena, podemos crear una clase como la siguiente:

```
public class TareaParalelizada extends SwingWorker<String, Integer>{
    /* Barra de progreso que la tarea irá actualizando a medida
    que los cálculos progresen */
    JProgressBar pbBarraProgreso;

    /*Intervalo de números que se van a sumar*/
    int min, max;

    /*Constructor*/
    public TareaParalelizada(JProgressBar pbBarraProgreso, int min, int max) {
        this.pbBarraProgreso = pbBarraProgreso;
        this.min = min;
        this.max = max;
    }

    @Override
    protected String doInBackground() throws Exception {
        /*Avisamos a la barra de cuales son los valores más pequeños
        y más grandes que se van a recorrer */
        this.pbBarraProgreso.setMaximum(max);
        this.pbBarraProgreso.setMinimum(min);
        int suma=0;

        /*Recorremos los números...*/
        for (int i=0; i<2000; i++){
            suma=suma+i;
            Thread.sleep(1);
            /*...y actualizamos la barra para que el usuario pueda
            ir viendo el progreso*/
            this.pbBarraProgreso.setValue(i);
        }
        return "Sumado:"+suma;
```

```
    }
}
```

Y despues asociemos el siguiente código al gestor del evento `click` para el `btnBotonConHilos`.

```
private void btnConHilosMouseClicked(java.awt.event.MouseEvent evt) {
    TareaParalelizada t=new TareaParalelizada(this.pbProgreso, 1, 2000);
    t.execute();
}
```

si ahora repetimos la ejecución de la aplicación pero ahora pulsamos el boton «Con hilos» veremos que **SÍ PODEMOS INTERACTUAR CON EL ÁREA DE TEXTO** . De hecho podríamos hacer cualquier otra cosa y la vez ir visualizando el progreso de la tarea, lo cual es claramente mucho más recomendable.