

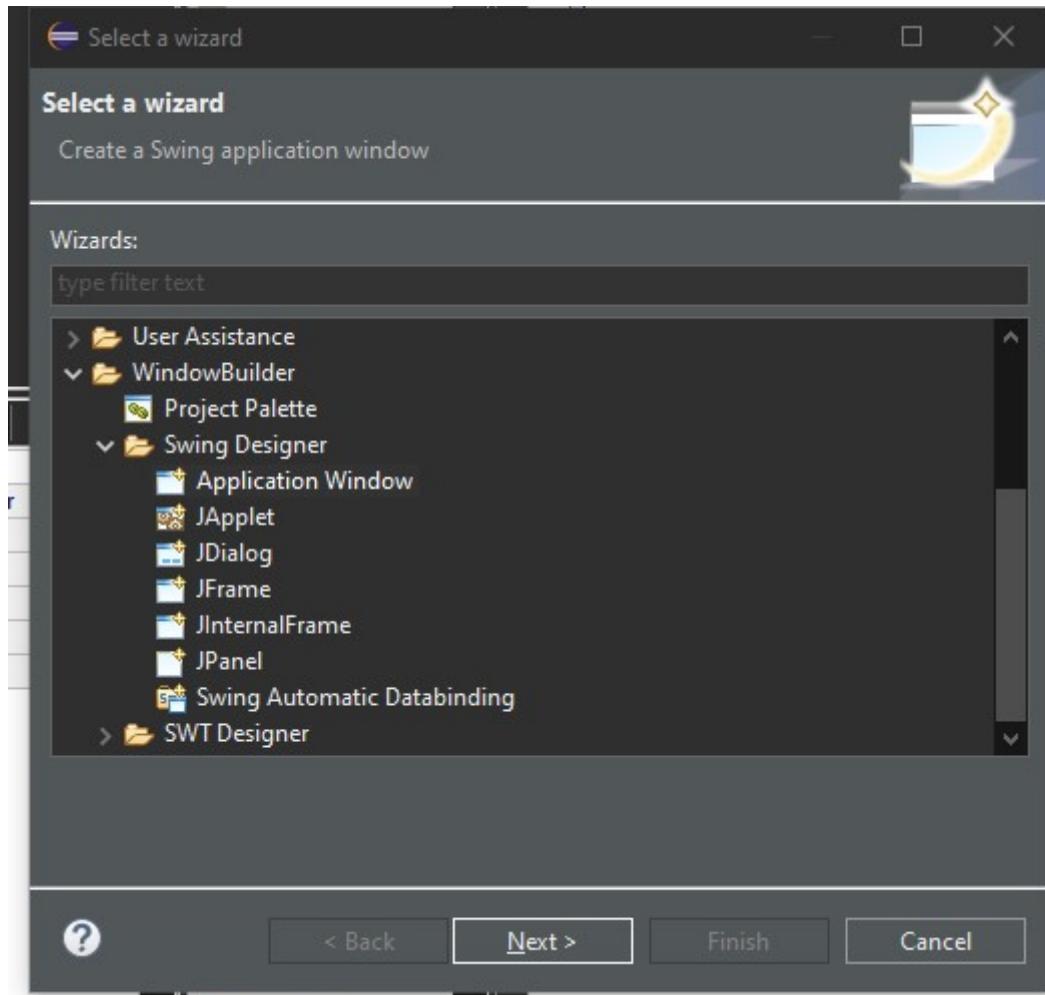
JTable

Vamos a implementar una JTable que muestre los hilos lanzados con un contador de iteraciones por Hilo y otro contador de iteraciones totales:

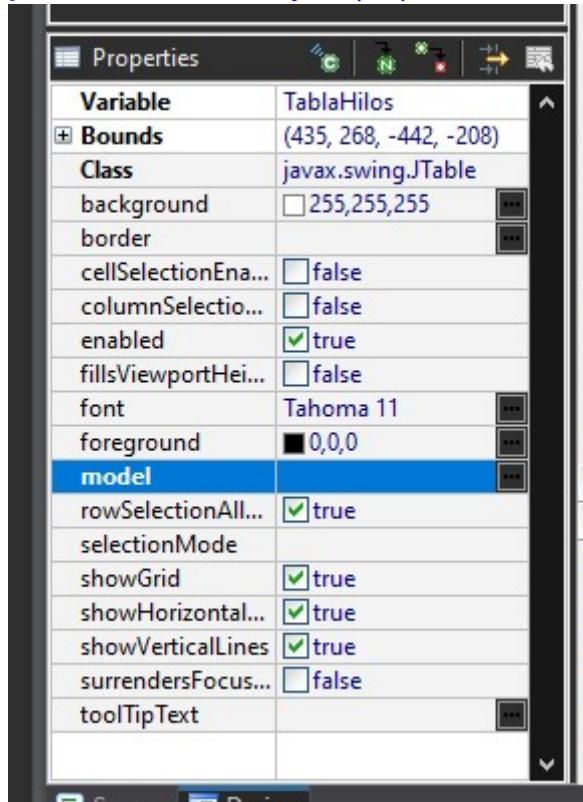


Implementación de la GUI

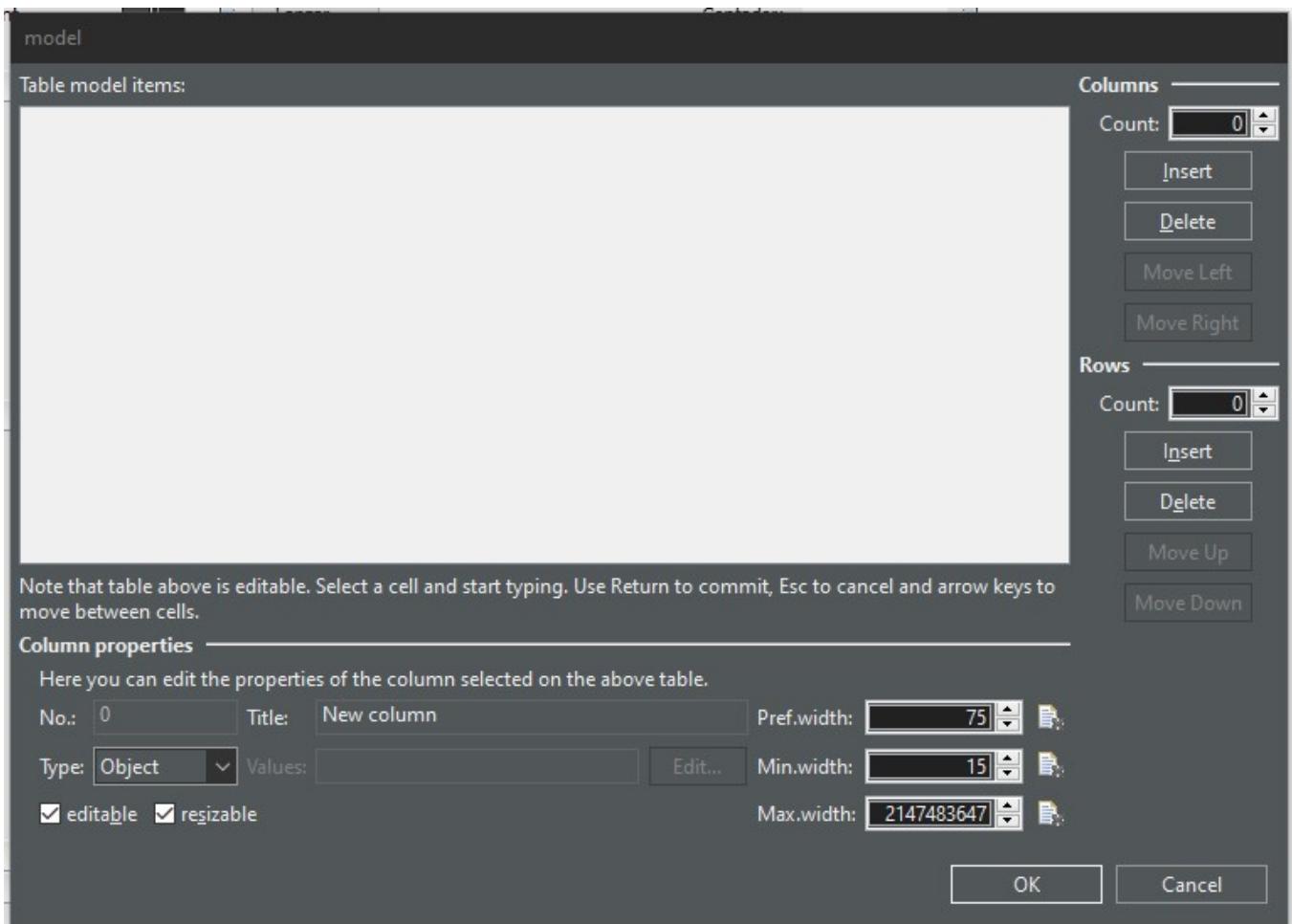
Primero creamos una nueva ventana desde el Window Builder:



A continuación insertamos la JTable en la ventana y en propiedades de esta vamos a model:



Cuando le demos a model nos saldrá la siguiente ventana con el editor de la tabla:



En la derecha le daremos a insertar una vez, colocaremos abajo el nombre de la columna, en este caso conteo, le daremos a insertar de nuevo y pondremos como nombre Hilos. Podemos ajustar el tamaño de las columnas abajo, en este caso la columna Hilos será de tamaño 200:

model

Table model items:

Hilos	Conteo
-------	--------

Columns

Count: 2

Rows

Count: 0

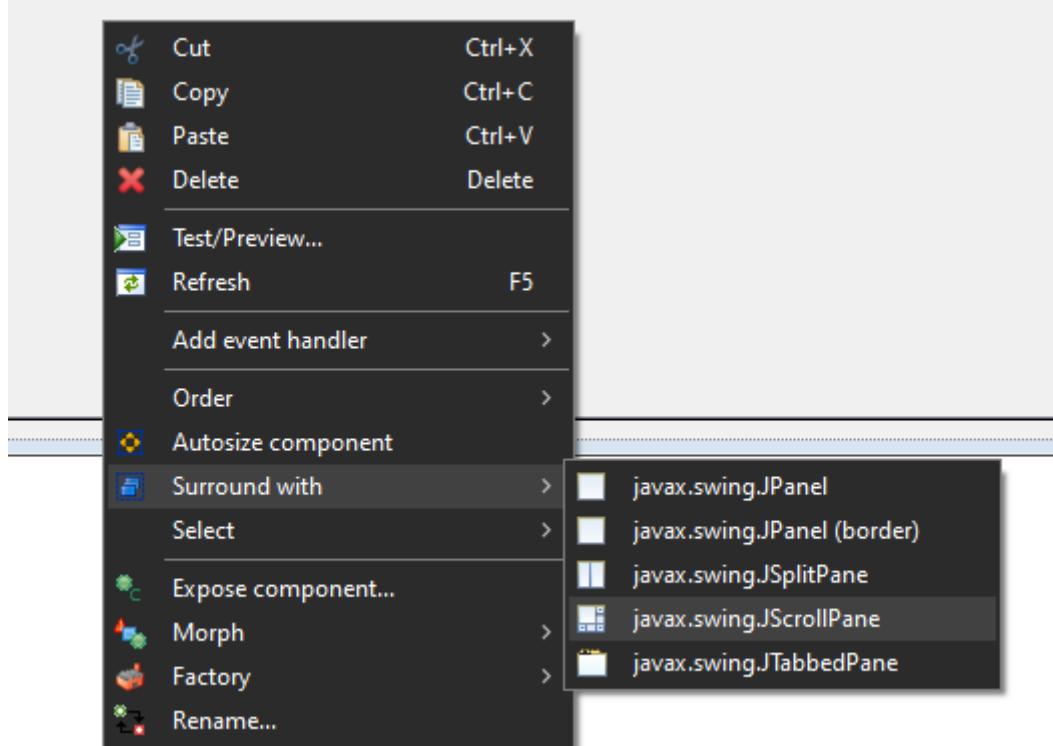
Note that table above is editable. Select a cell and start typing. Use Return to commit, Esc to cancel and arrow keys to move between cells.

Column properties

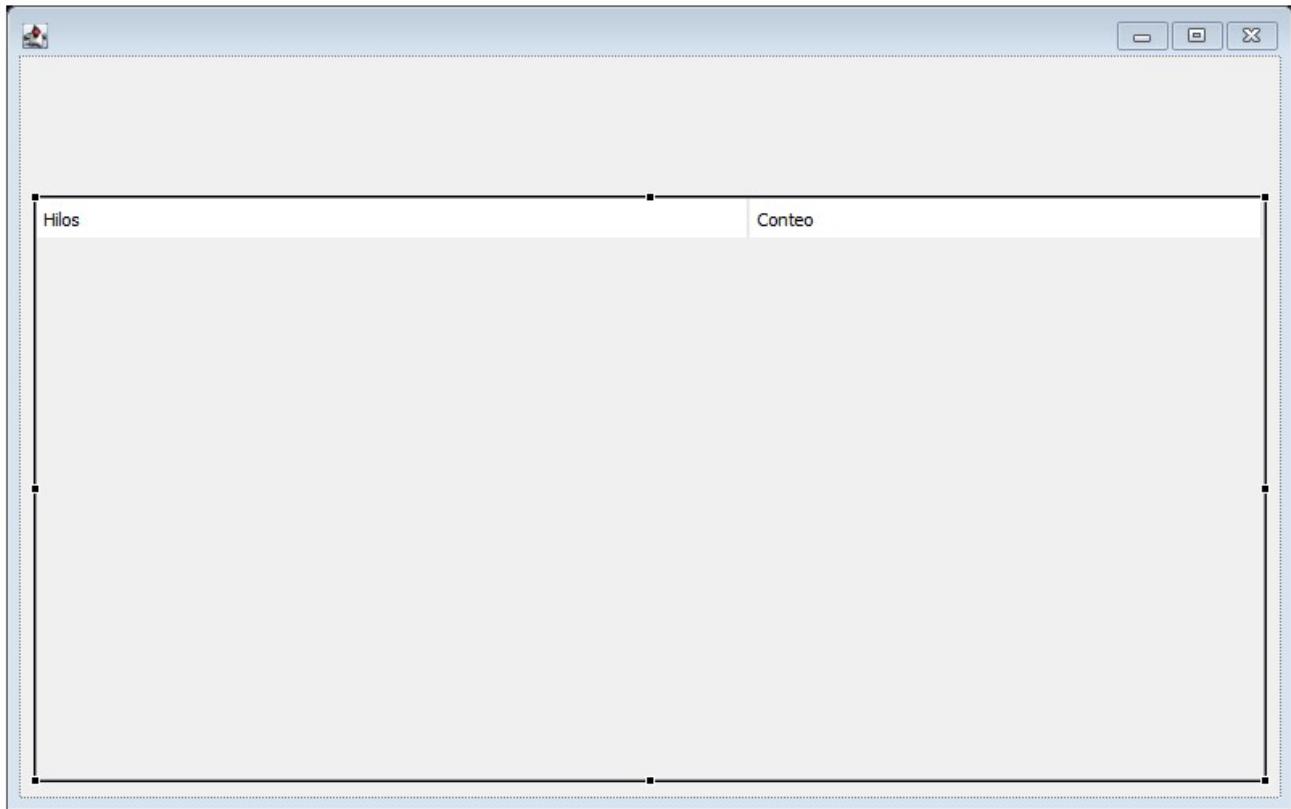
Here you can edit the properties of the column selected on the above table.

No.: 0	Title: Hilos	Pref.width: 200
Type: Object	Values:	Min.width: 15
<input checked="" type="checkbox"/> editable <input checked="" type="checkbox"/> resizable		Max.width: 2147483647

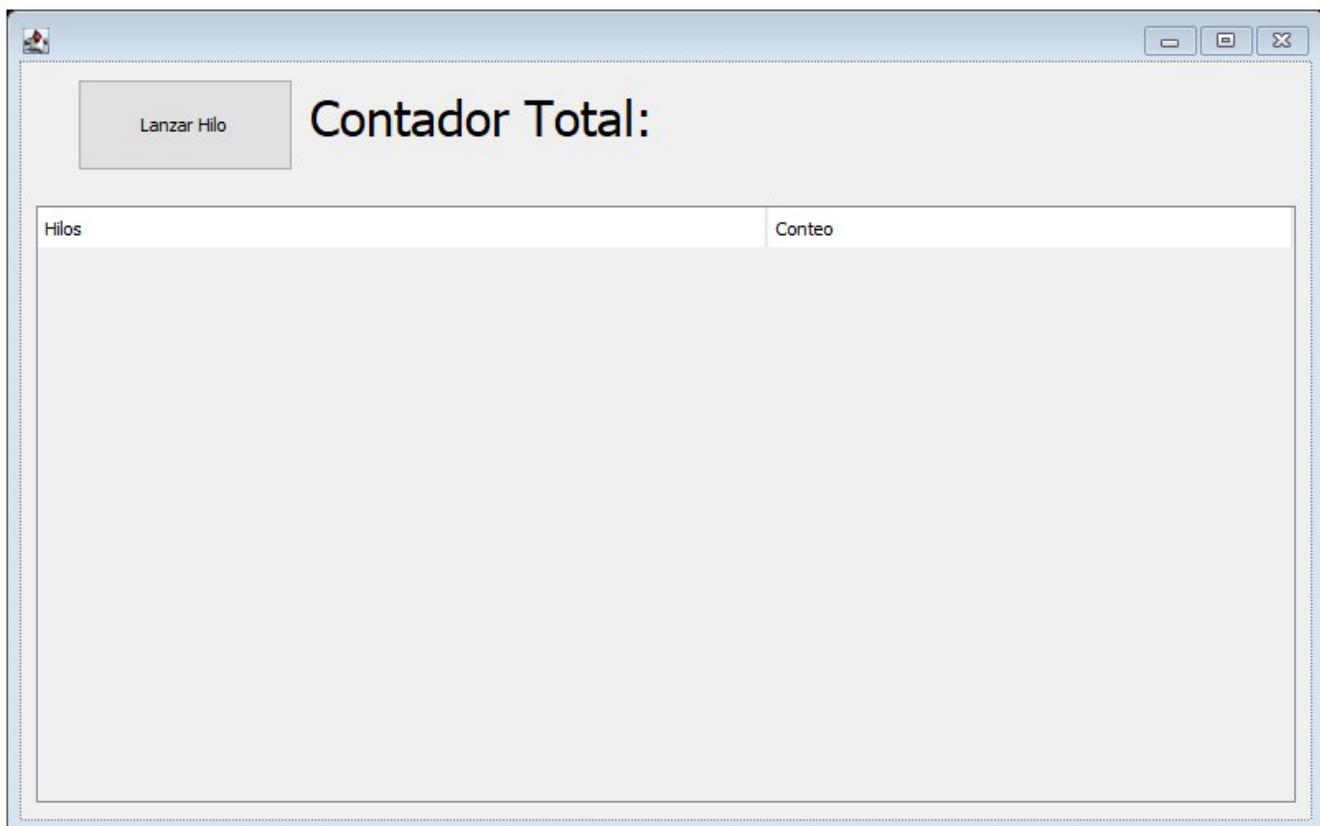
Tras darle a aceptar, no veremos nada en donde hemos insertado la tabla, ahora le daremos click derecho al recuadro donde debería estar la tabla e iremos a Surround With y seleccionaremos JScrollPane:



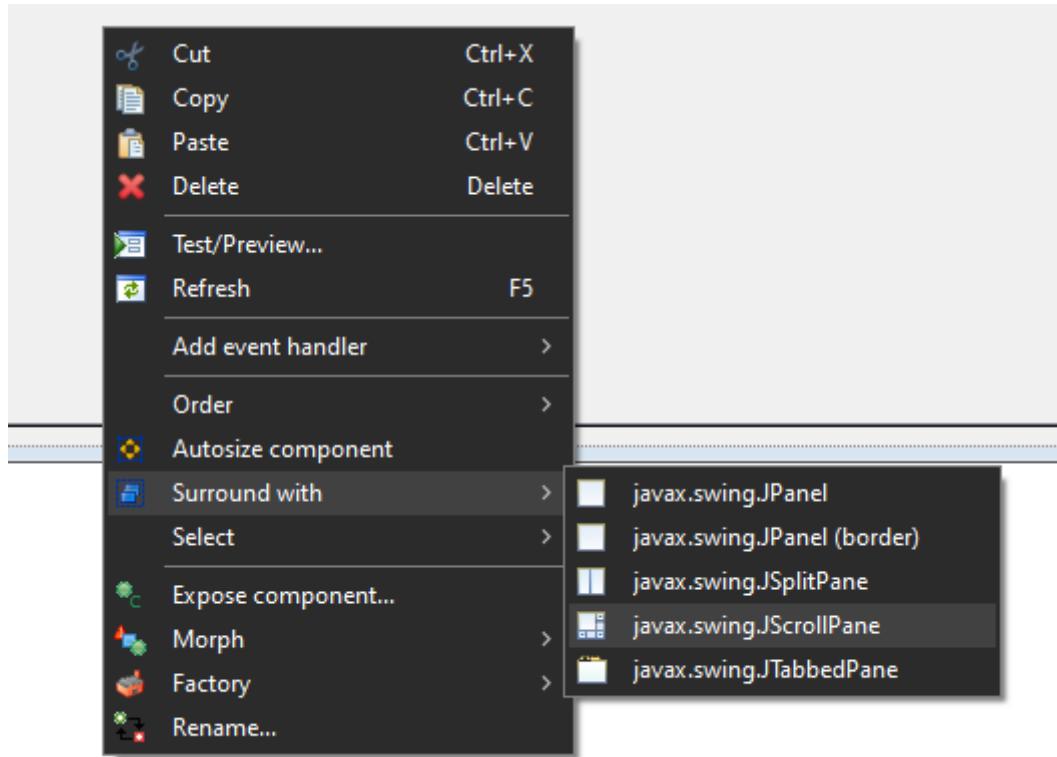
Tras eso tendremos algo como esto:



Ahora que hemos terminado con la inserción de la JTable insertaremos el Botón Lanzar y la JLabel que irá al lado del contador de iteraciones totales:

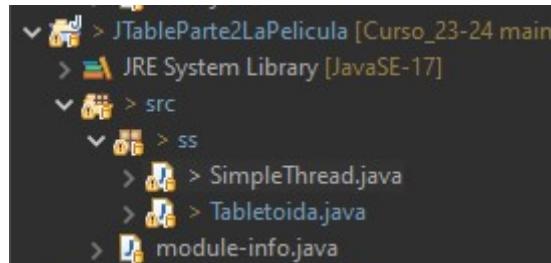


Finalmente insertaremos un JTextField no editable para que muestre los conteos totales:



Implementación Funcional Simple

Estructura Clases



Variables SimpleThread

```
public class SimpleThread extends Thread{//Extendemos la clase Thread
    public static final int FOR_EVER = -1;//Constante para ejecución sin fin
    protected long delay;//variable para el retraso de ejecución
    protected int times;//Variable para el número de repeticiones
    protected int lecontadorInterno;//Variable para contar las iteraciones de este hilo
```

Variables Tabletoida

Hilos

Para implementar los hilos que lanzaremos crearemos una nueva clase a la que llamaremos SimpleThread:

```

1 package ss;
2
3 public class SimpleThread extends Thread{//Extendemos la clase Thread
4     public static final int FOR_EVER = -1;//Constante para ejecución sin fin
5     protected long delay;//variable para el retraso de ejecución
6     protected int times;//Variable para el número de repeticiones
7     //Constructor del hilo, indicamos el retraso que tendrá en re-ejecutarse
8     //y cuantas veces se volverá a ejecutar
9     public SimpleThread (long delay, int times) {
10         System.out.println("Constructor SimpleThread");//Texto a mostrar en consola cuando se construye un hilo
11         this.delay = delay;
12         this.times = times;
13     }
14     public void run() {
15         try {
16             for(int aux = times; (times>=0)|| (aux == FOR_EVER); times--) {
17                 System.out.println("Mi Delay es:"+delay);//Mostramos en consola el delay
18                 sleep(delay);//Para la ejecución por el tiempo de retardo especificado
19             }
20         }catch(Exception e) {
21             System.out.println("Error.");
22         }
23     }
24
25
26 }
27

```

Tras esto podemos ir añadiendo la funcionalidad al botón Lanzar para que pueda lanzar hilos:

```

JButton botonLanzar = new JButton("Lanzar Hilo");
botonLanzar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) { //-----Botón Lanzar (EN CONSTRUCCIÓN)
        SimpleThread hilillo = new SimpleThread(5000, 5); //nuevo objeto tipo hilo
        hilillo.start(); //Lanzamos nuevo hilo
        System.out.println("Hilo" + hilillo + "lanzado");
        contTotal++;
        contadorTotalGui.setText(""+ contTotal + ""); //cambiamos valor mostrado en JTextField, esto necesita ser modificado
    }
});
botonLanzar.setBounds(35, 11, 131, 56);
getContentPane().add(botonLanzar);

```

Botón Lanzar

Ahora haremos que cada vez que pulsamos el botón cada hilo que arranquemos se vaya almacenando en una lista de hilos y que tras eso, vuelque esta lista de hilos a la Jtable para Rellenarla. A su vez, se sumará el valor de los campos que almacenen el conteo de iteraciones de cada hilo en una variable cuyo valor se escribirá en el JTextField que mantiene el conteo total. Hasta aquí hemos implementado la funcionalidad de forma que la tabla se actualiza cada vez que pulsamos el botón lanzar, pero no se actualiza automáticamente.

```

 JButton botonLanzar = new JButton("Lanzar Hilo");
 botonLanzar.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) { //<-----Botón Lanzar (EN CONSTRUCCIÓN)
         simpleThread hilillo = new SimpleThread(500, 5); //nuevo objeto tipo hilo
         hilillo.start(); //Lanzamos nuevo hilo
         auxThread = ("ID: " + hilillo); //Almacenamos el identificador del hilo en la variable auxThread
         contTotal = 0;
         System.out.println("Hilo " + auxThread + " lanzado");

         try {
             listaHilos.add(hilillo); //añadimos el hilo a las lista de Hilos
         } catch (Exception E){
             System.out.println("Error añadiendo el hilo"+ auxThread +"A la lista de hilos");
         }

         DefaultTableModel model = (DefaultTableModel) tHilos.getModel(); //Pillamos el modelo de la tabla para modificar filas y columnas
         model.setRowCount(0); //Vaciamos la tabla
         for(int i=0; i< listaHilos.size(); i++) { //Actualizamos la table la tabla
             model.addRow(new Object[]{" "+ listaHilos.get(i)+"", ""+listaHilos.get(i).lecontadorInternos+""}); //añadimos Fila a la tabla
             contTotal = contTotal + listaHilos.get(i).lecontadorInternos; //sumamos el valor de los conteos en la variable contTotal
             contadorTotalgui.setText(" " + contTotal + ""); //cambiamos valor mostrado en JTextField, esto necesita ser modificado
         }
     }
 });

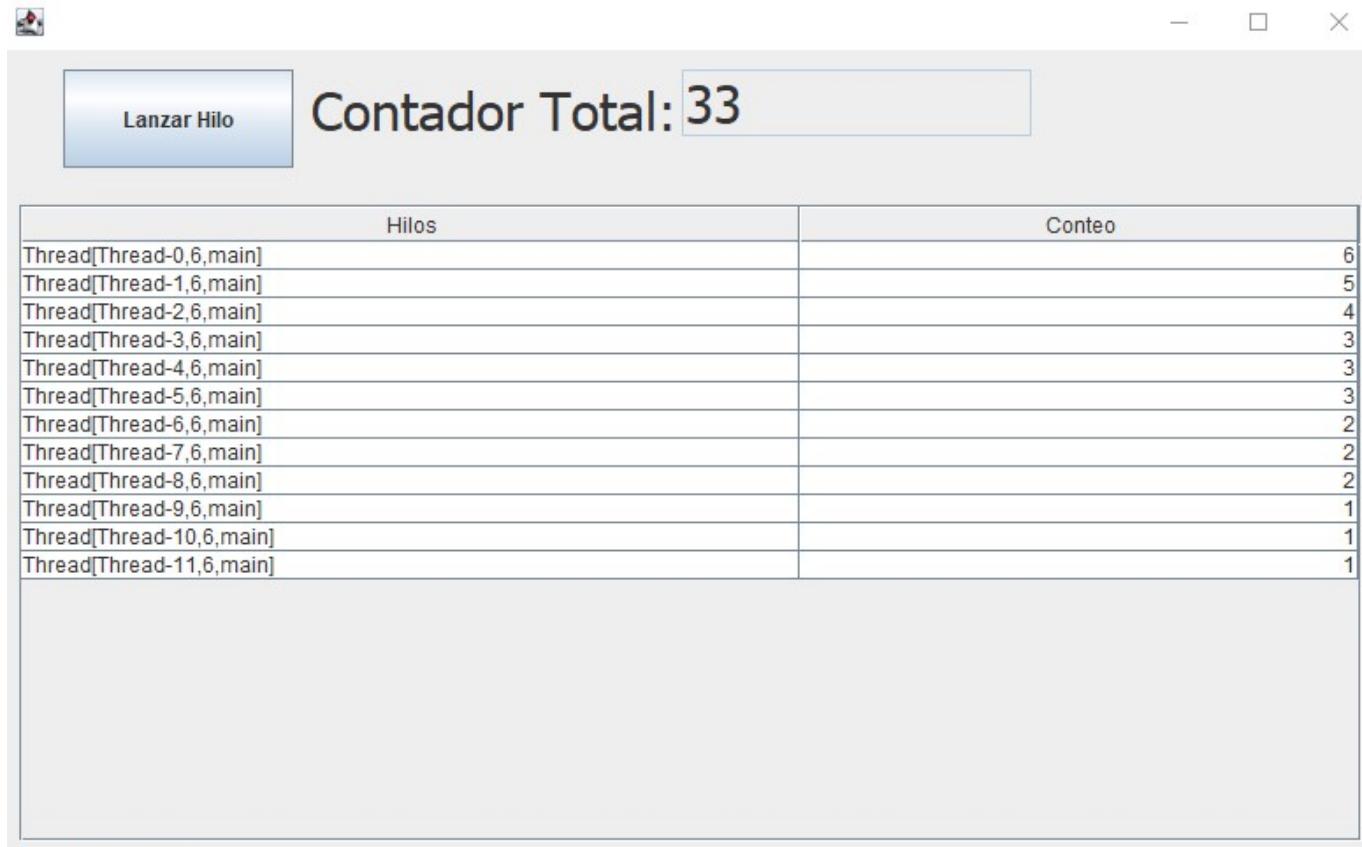
 botonLanzar.setBounds(35, 11, 131, 56);
 getContentPane().add(botonLanzar);

```

Resultado

Ahora la tabla se verá de la siguiente forma:





The screenshot shows a Java Swing application window. At the top left is a button labeled "Lanzar Hilo". To its right is a label "Contador Total: 33". Below these are two buttons: a red one with a minus sign and a green one with a plus sign. The main area is a JTable with two columns: "Hilos" and "Conteo". The data in the table is as follows:

Hilos	Conteo
Thread[Thread-0,6,main]	6
Thread[Thread-1,6,main]	5
Thread[Thread-2,6,main]	4
Thread[Thread-3,6,main]	3
Thread[Thread-4,6,main]	3
Thread[Thread-5,6,main]	3
Thread[Thread-6,6,main]	2
Thread[Thread-7,6,main]	2
Thread[Thread-8,6,main]	2
Thread[Thread-9,6,main]	1
Thread[Thread-10,6,main]	1
Thread[Thread-11,6,main]	1

Sobre Actualizar la Tabla Automáticamente

Para actualizar la tabla automáticamente deberíamos pasar por referencia la tabla a la clase SimpleThread y utilizar el método: miTableModel.fireTableDataChanged().

Implementación Funcional Compleja

Ahora veremos una implementación alternativa algo más compleja. Utilizaremos la misma GUI que en la anterior implementación, pero cambiarán la funcionalidad y clases.

Clase ModeloTabla

Primero implementaremos esta clase para lo que sería el model de la JTable. Aquí implementaremos métodos para contar las filas y columnas actuales de la tabla, además de un método para ir obteniendo los datos desde la lista de hilos a partir de la fila y columna en la que estemos para llenar la tabla.

```
1 package ss;
2
3 import ss.Tabletoida;
4 import ss.SimpleThread;
5 import javax.swing.table.DefaultTableModel;
6
7 public class ModeloTabla extends DefaultTableModel{
8     private Tabletoida ventana = null;//Objeto de tipo Tabletoida, invoca la ventana principal
9
10    public ModeloTabla(Tabletoida ventana) {//constructor del modelo
11        this.ventana = ventana;
12    }
13
14    @Override
15    public int getColumnCount() {
16        return this.ventana.getColName().length;//Contamos cuantas columnas tiene la tabla
17    }
18
19    @Override
20    public String getColumnName(int column) {//Obtenemos el nombre de la columna indicada
21        return this.ventana.getColName()[column];
22    }
23
24    @Override
25    public int getRowCount() {//contamos cuantas filas hay
26        int rowCount=0;
27        if(this.ventana != null) {
28            rowCount = this.ventana.getListaHilos().size();
29        }
30        return rowCount;
31    }
32
33    @Override
34    public Object getValueAt(int row, int column) {//Obtener el valor almacenado en la lista que corresponde a la posición indicada
35        Object returnValue = null;
36        if (this.ventana.getListaHilos().size()>0) { // Si hay hilos insertados en la lista
37            SimpleThread hilo = (SimpleThread)this.ventana.getListaHilos().get(row);//declaramos hilo con el valor del que se encuentra en cierta fila "row"
38            switch (column) {//En función de la columna que queramos obtener (0 para el hilo y 1 para el conteo)
39                case 0://Obtenemos ID de Hilo
40                    returnValue = hilo.getId();
41                    break;
42                case 1://Obtenemos Valor del contador del hilo en cuestión
43                    returnValue = hilo.getContadorInterno();
44                    break;
45                default://en caso de solicitar otra cosa, no se devuelve nada
46                    returnValue = "";
47            }
48        }
49        return returnValue;//Devolvemos el elemento seleccionado.
50    }
51
52    public boolean isCellEditable(int row, int column) {//Hacemos que no se puedan editar las celdas de la tabla
53        return false;
54    }
55
56 }
```

Clase AccionBoton

```
1 package ss;
2
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 import java.util.Iterator;
6 import java.util.List;
7
8 public class AccionBoton implements ActionListener{
9     private List<Thread> hilos;
10    private ModeloTabla model;
11
12    public AccionBoton(List <Thread> hilos, ModeloTabla model) {
13        this.hilos = hilos;
14        this.model = model;
15    }
16
17    public void actionPerformed(ActionEvent e) { //acción a ejecutar cuando se pulsa el botón
18        if (this.hilos != null) { //Si la lista de hilos no está vacía
19            Iterator<Thread> it = this.hilos.iterator(); //Variable para iterar por las posiciones de la tabla
20            while(it.hasNext()) { //mientras hayan hilos en la siguiente posición de la lista
21                Thread hilo = (Thread)it.next(); //avanzamos a la siguiente posición de la lista de hilos
22                hilo.start(); //arrancamos el hilo
23            }
24            this.model.fireTableDataChanged(); //invocamos una actualización de tabla
25        }else {
26            System.out.println("La lista de hilos está vacía...");
27        }
28    }
29
30    public List<Thread> getHilos(){ //método para devolver la lista de hilos
31        return this.hilos;
32    }
33 }
```

From:

<http://knoppia.net/> - Knoppia



Permanent link:

<http://knoppia.net/doku.php?id=dad:jtable>

Last update: **2023/09/29 10:06**