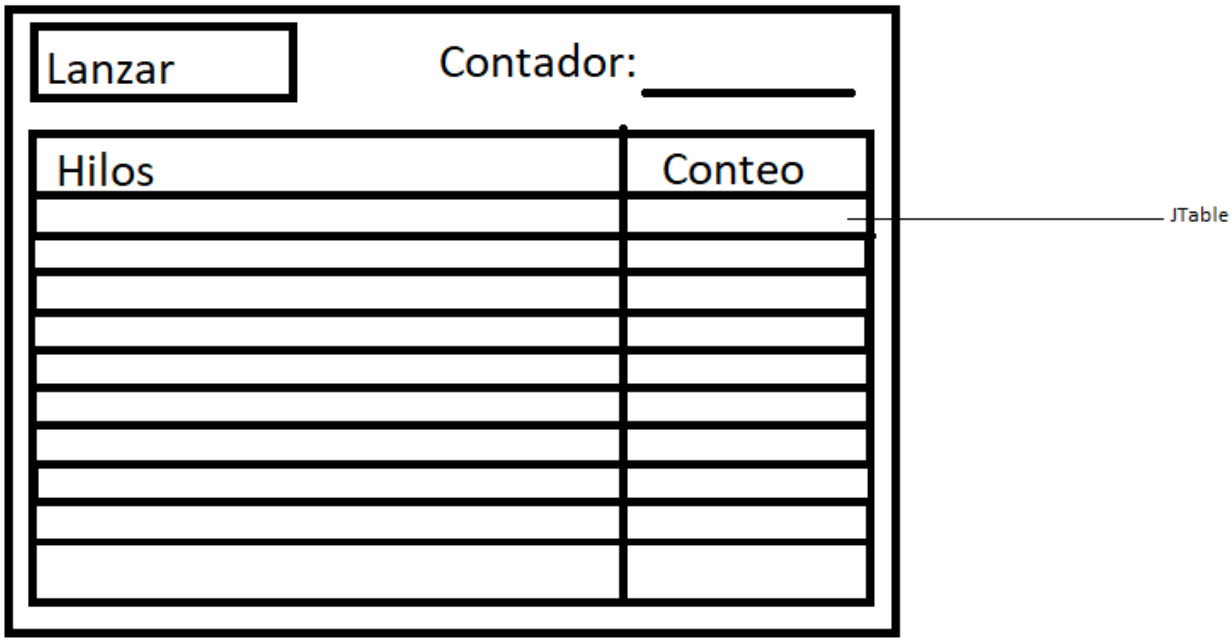
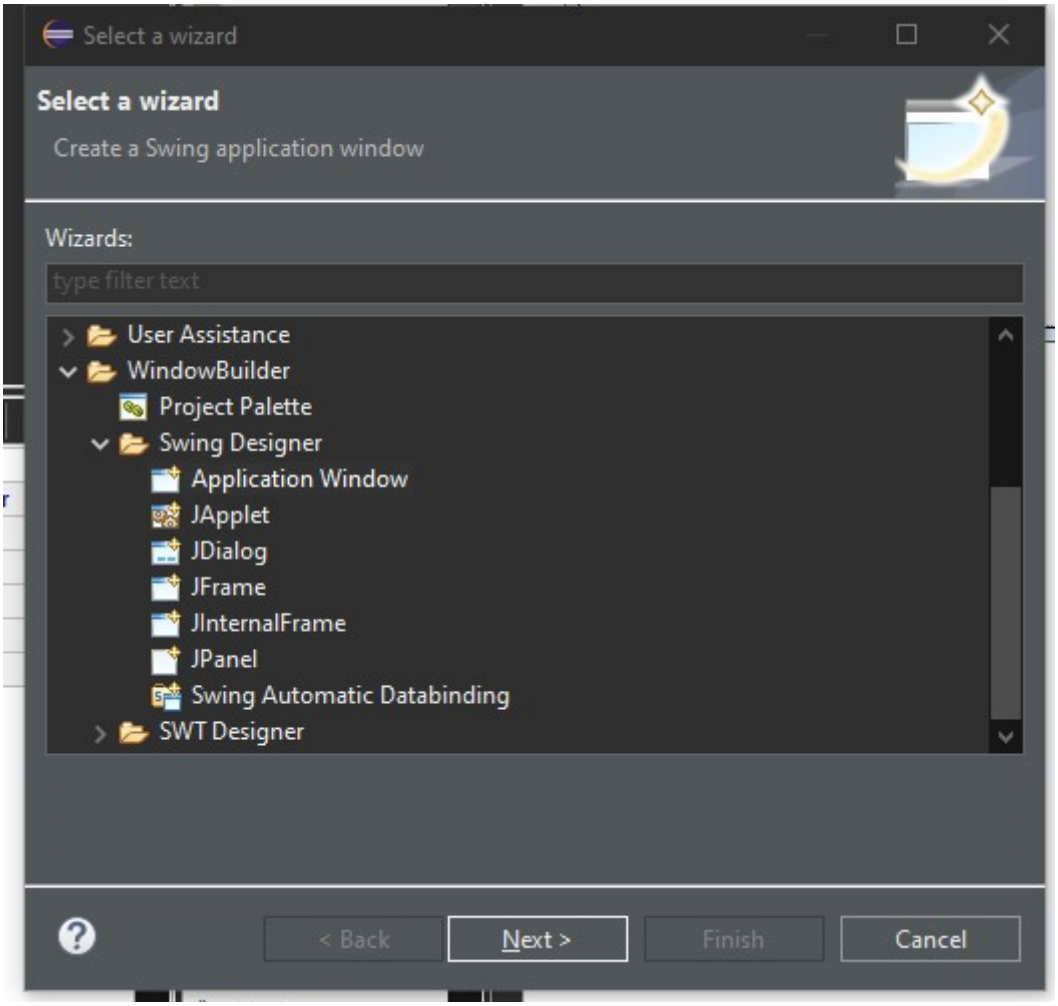
JTable

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

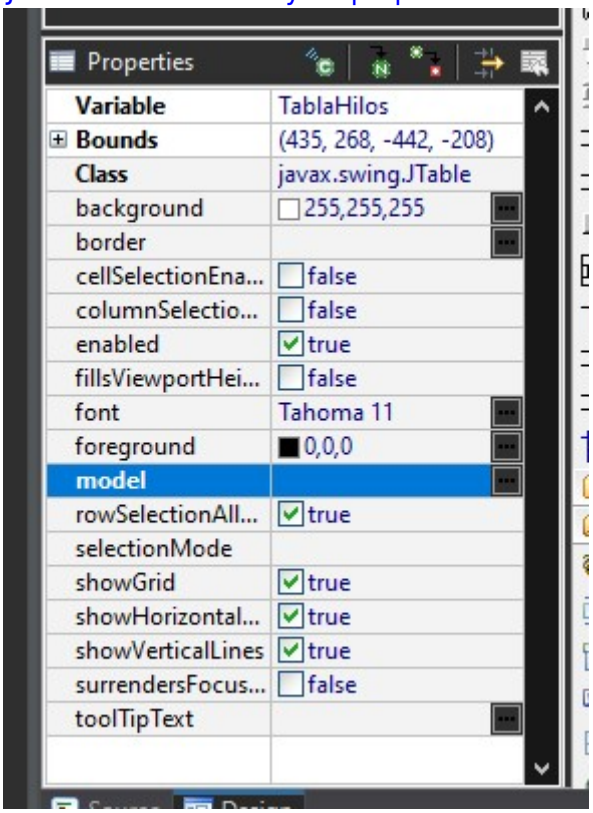


Implementación de la GUI

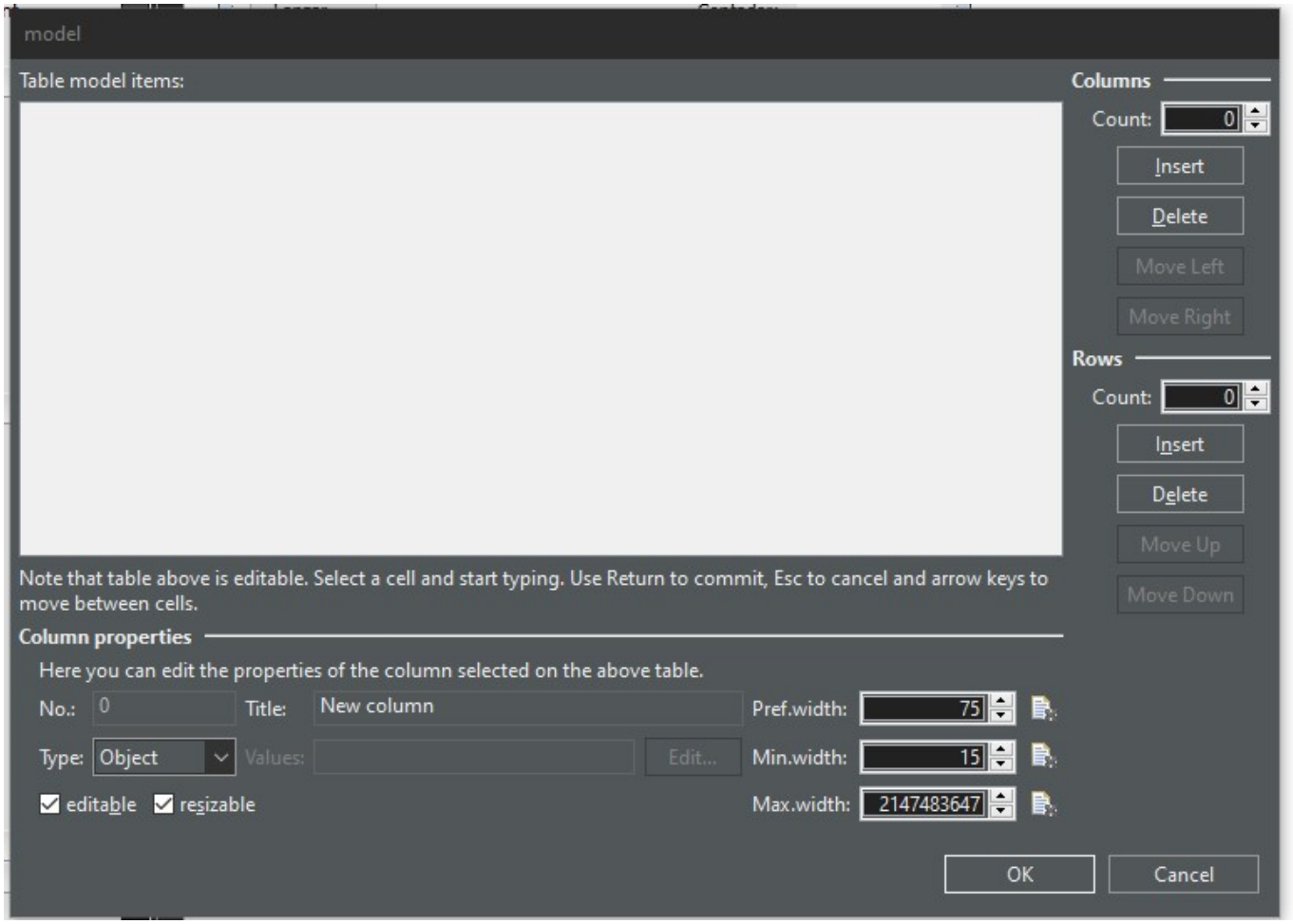
Primero creamos una nueva ventana desde el Window Builder:



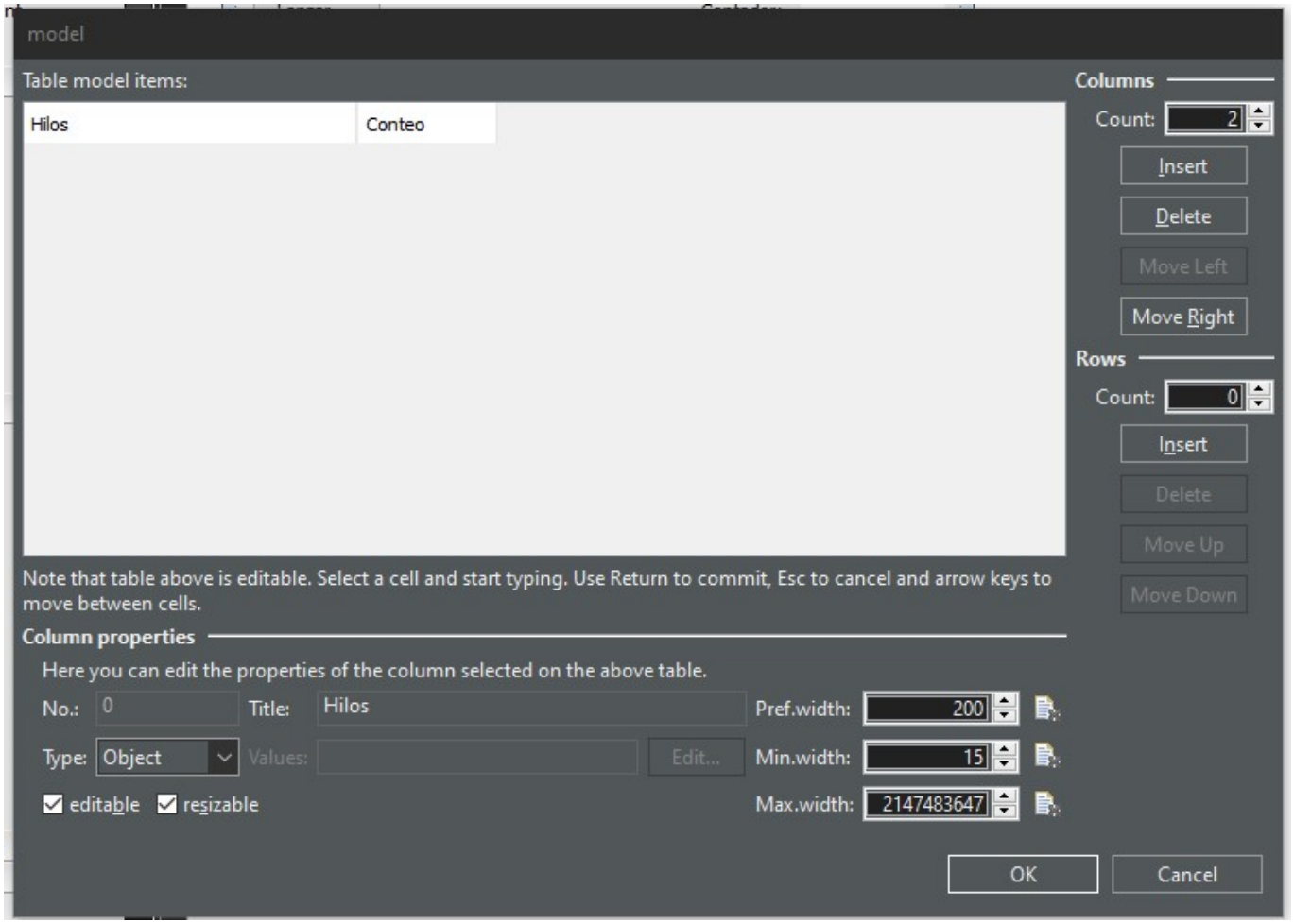
A continuación insertamos la JTable en la ventana y en propiedades de esta vamos a model:



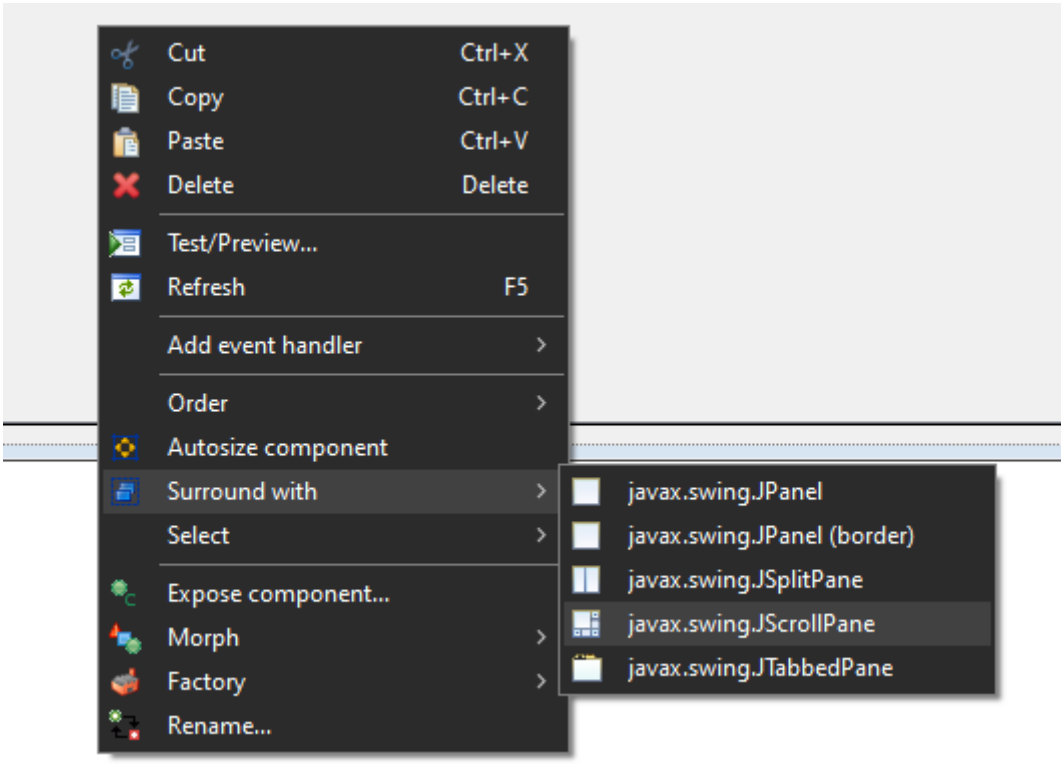
Cuando le demos a model nos saldrá la siguiente ventana con el editor de la tabla:



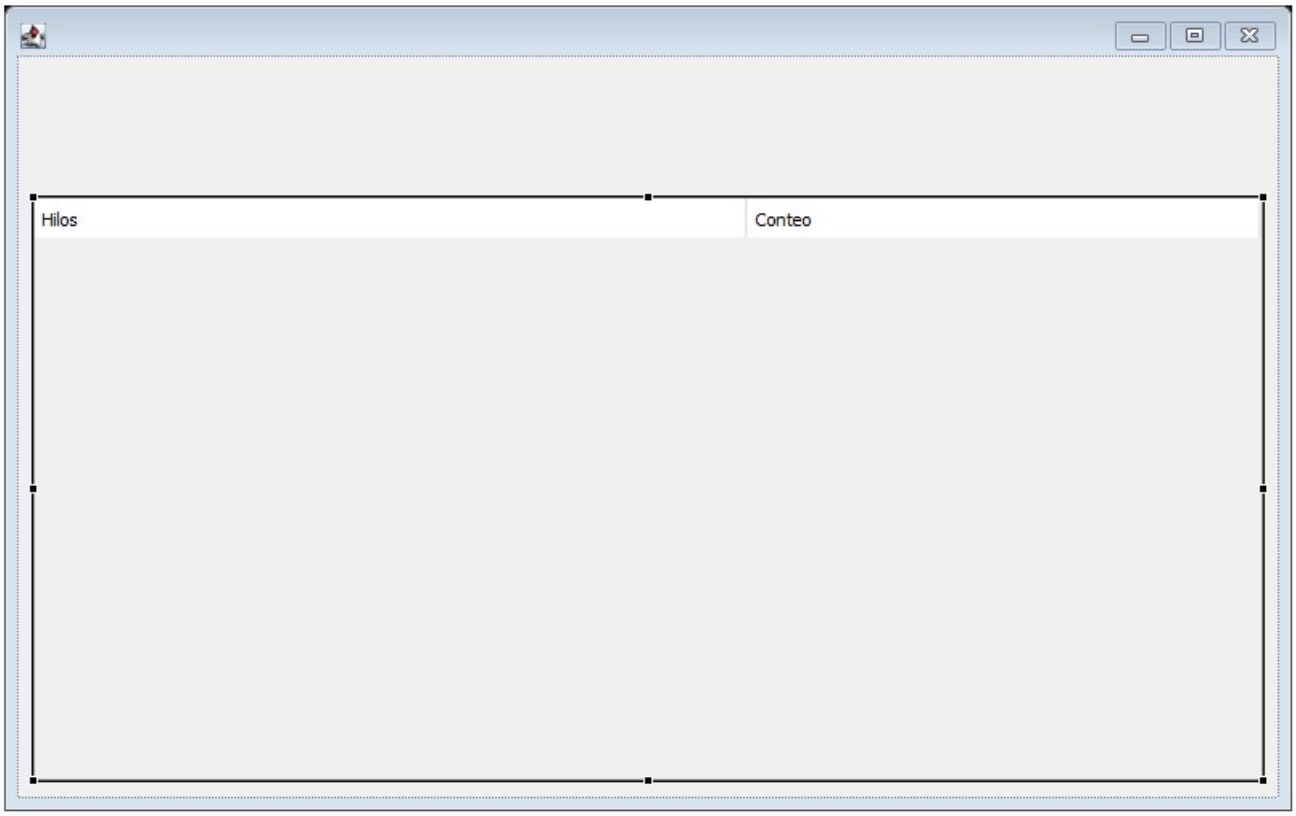
En la derecha le daremos a insertar una vez, colocaremos abajo el nombre de la columna, en este caso conteo, le daremos a insertar de nuevo y pondremos como nombre Hilos. Podemos ajustar el tamaño de las columnas abajo, en este caso la columna Hilos será de tamaño 200:



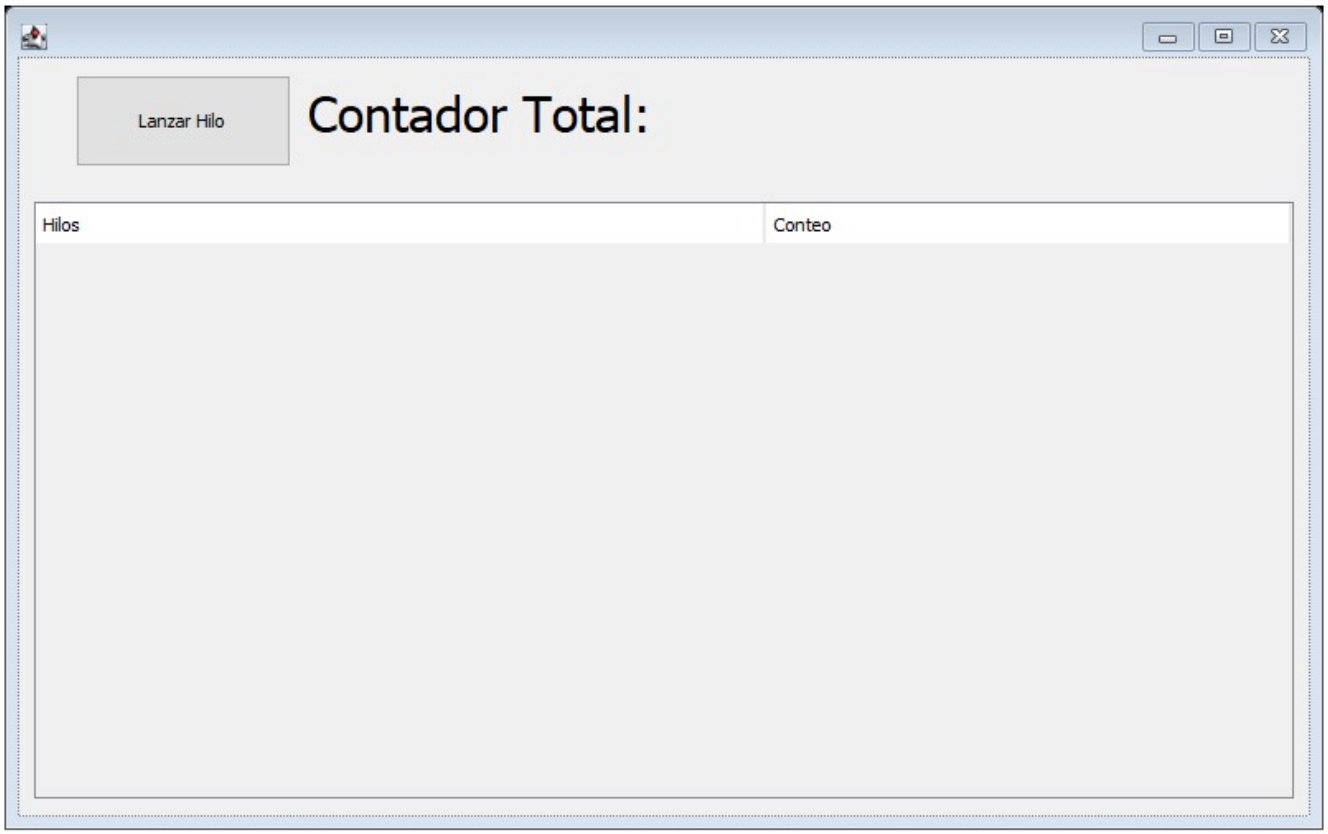
Tras darle a aceptar, no veremos nada en donde hemos insertado la tabla, ahora le daremos click derecho al recuadro donde debería estar la tabla e iremos a Surround With y seleccionaremos JScrollPane:



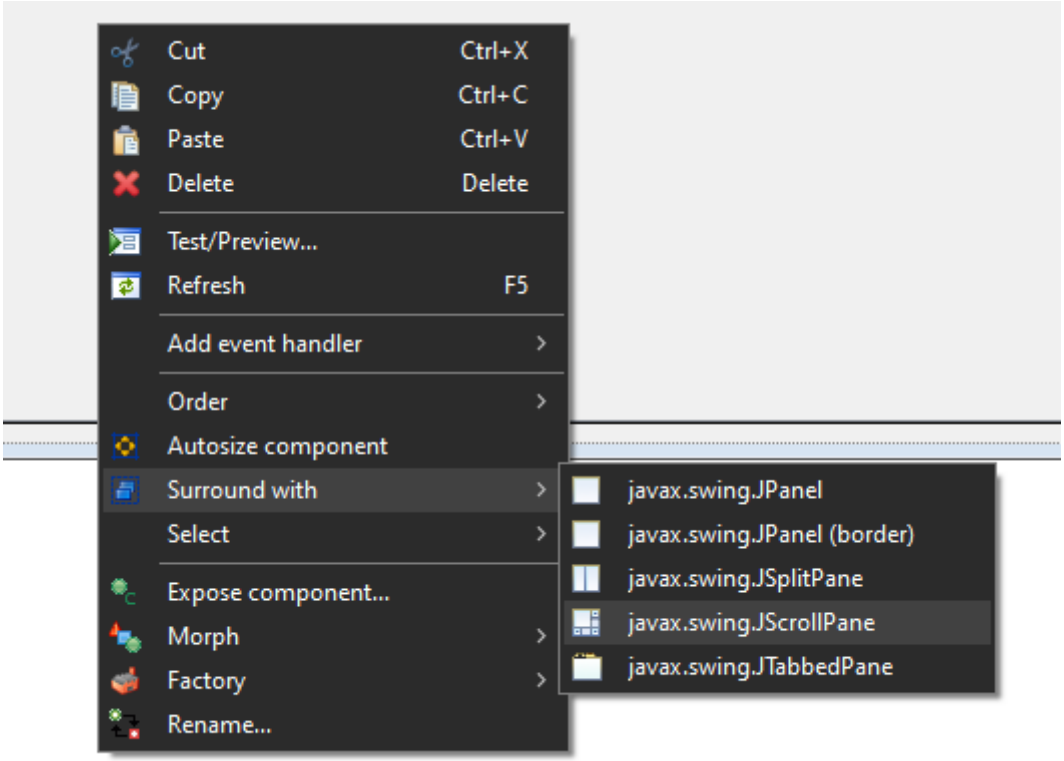
Tras eso tendremos algo como esto:



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:

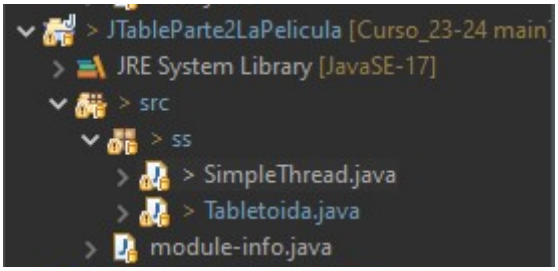


Finalmente insertaremos un JTextField no editable para que muestre los conteos totales:



Implementación Funcional

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 iteracionesa de este hilo
```

Variables Tabletoida

Hilos

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

```java
package ss;

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
    //Constructor del hilo, indicamos el retraso que trendrá en re-ejecutarse
    //y cuantas veces se volerá a ejecutar
    public SimpleThread (long delay, int times) {
        System.out.println("Constructor SimpleThread");//Texto a mostrar en consola cuando se construye un hilo
        this.delay = delay;
        this.times = times;
    }
    public void run() {
        try {
            for(int aux = times; (times>=0)||(aux == FOR_EVER); times--) {
                System.out.println("Mi Delay es:"+delay);//Mostramos en consola el delay
                sleep(delay);//Para la ejecución por el tiempo de retardo especificado
            }
        }catch(Exception e) {
            System.out.println("Error.");//En caso de que algo falle se mostrará este error
        }
    }


}
```

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

```java
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);
```

Tras eso haremos que cada vez que pulsemos 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

```java
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).lecontadorInterno+""});//añadimos Fila a la tabla
            contTotal = contTotal + listaHilos.get(i).lecontadorInterno;//sumamso 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);
```

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().

From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=dad:jtable&rev=1695896254

Last update: **2023/09/28 10:17**