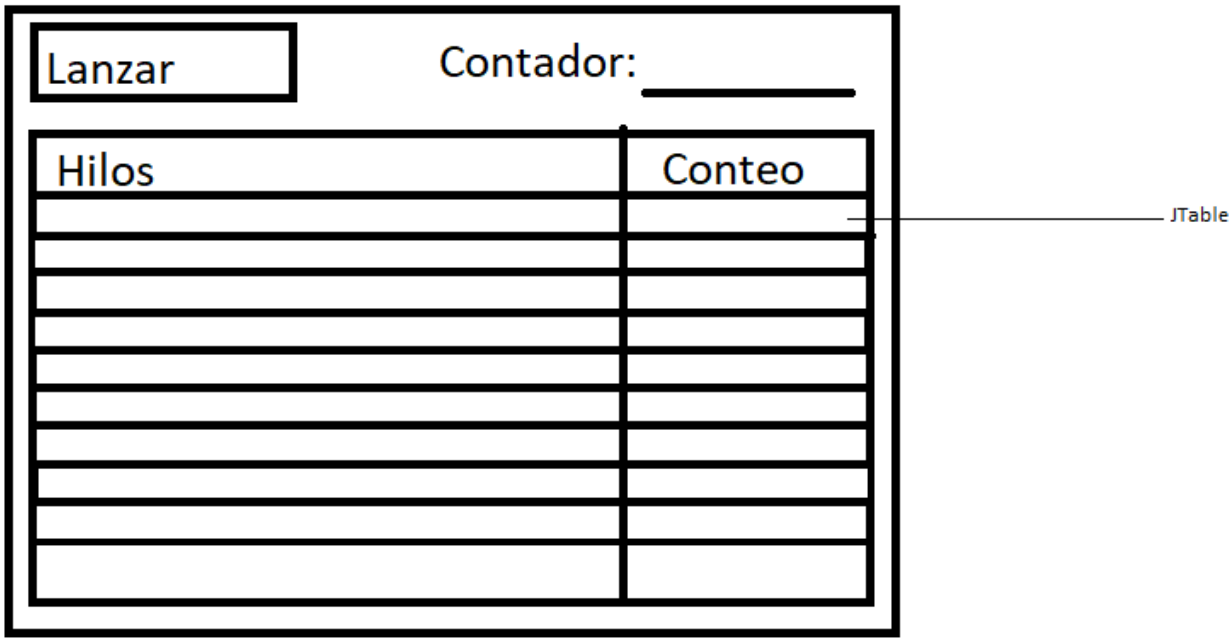
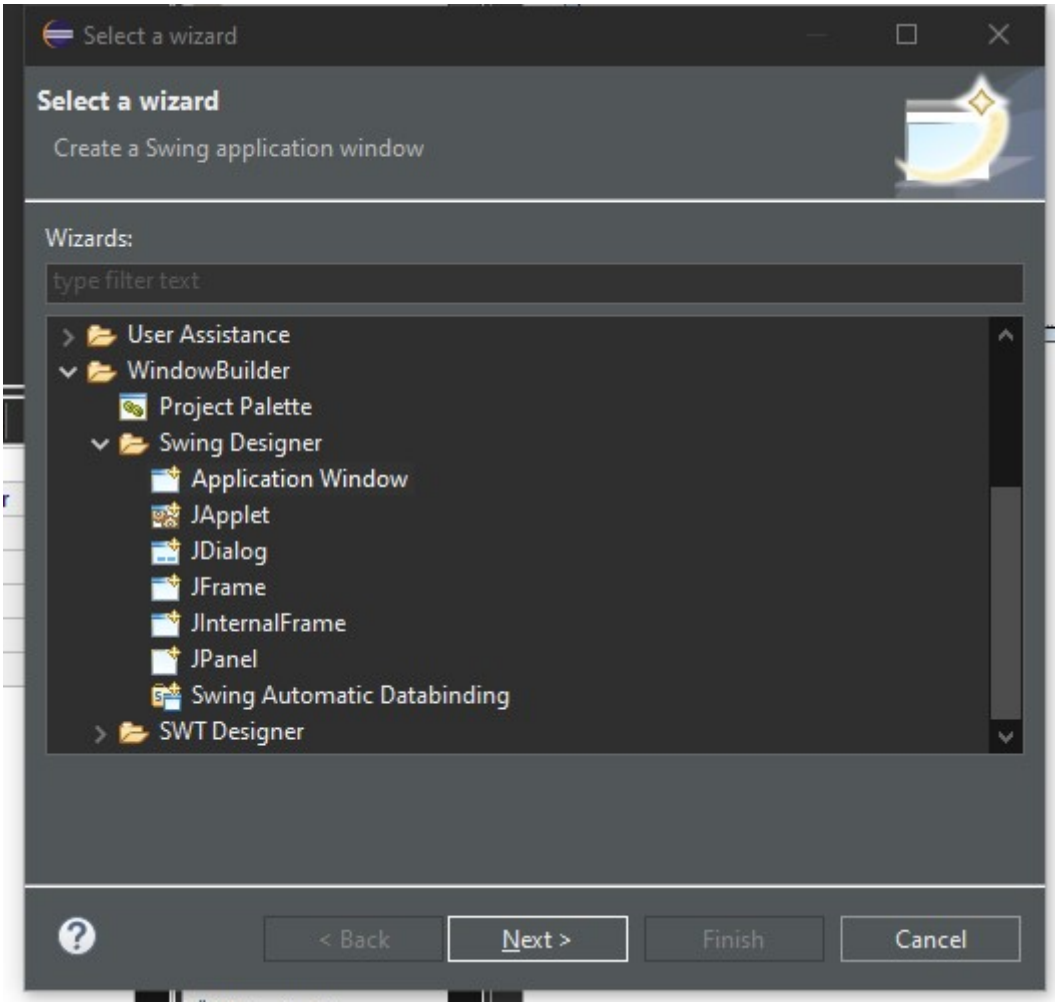
JTable

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

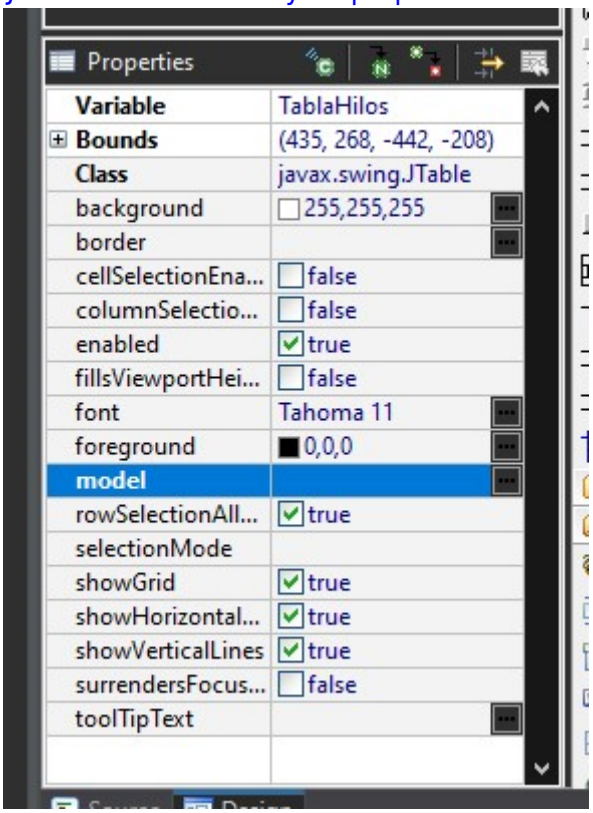


Implementación de la GUI

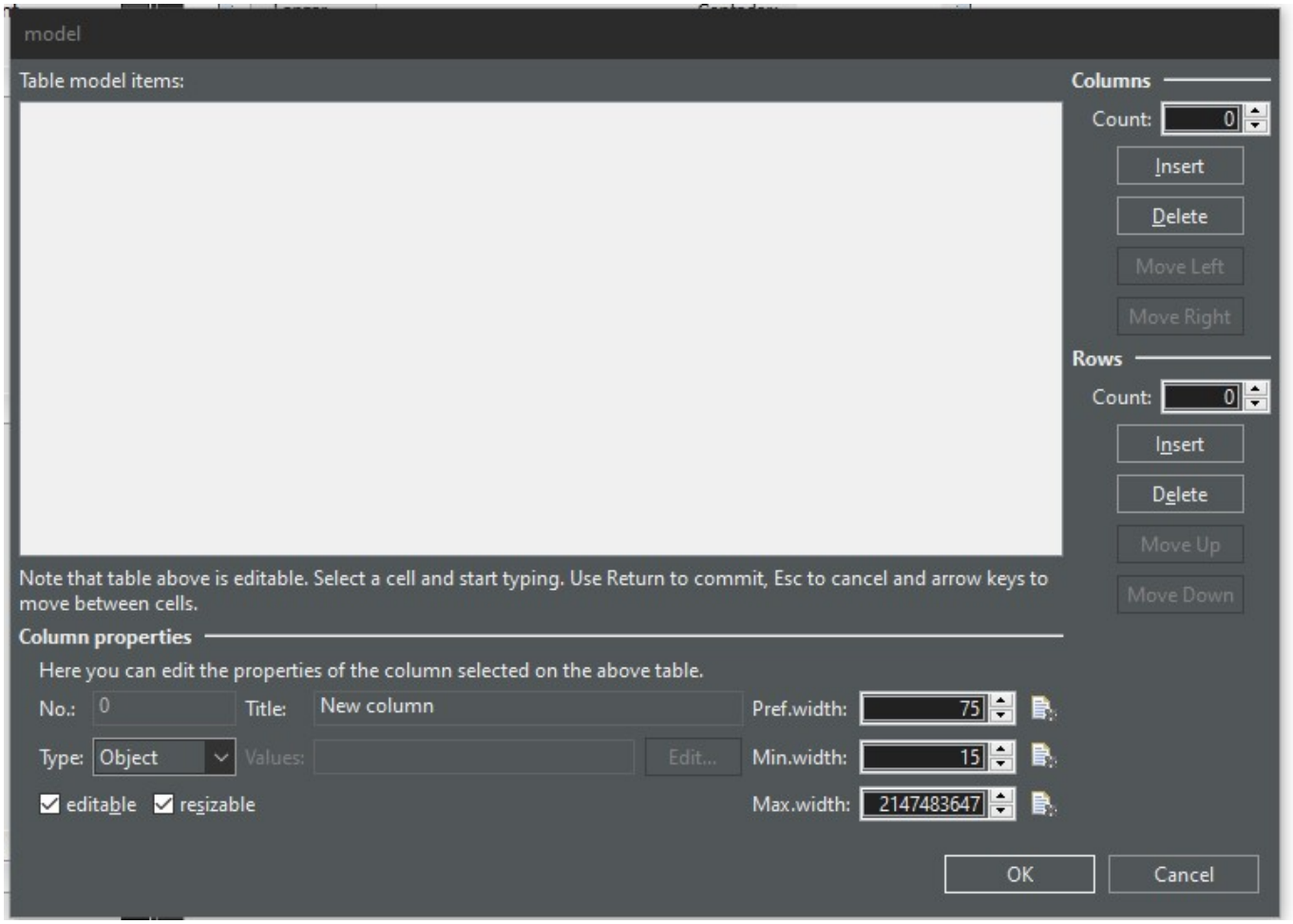
Primero creamos una nueva ventana desde el Window Builder:



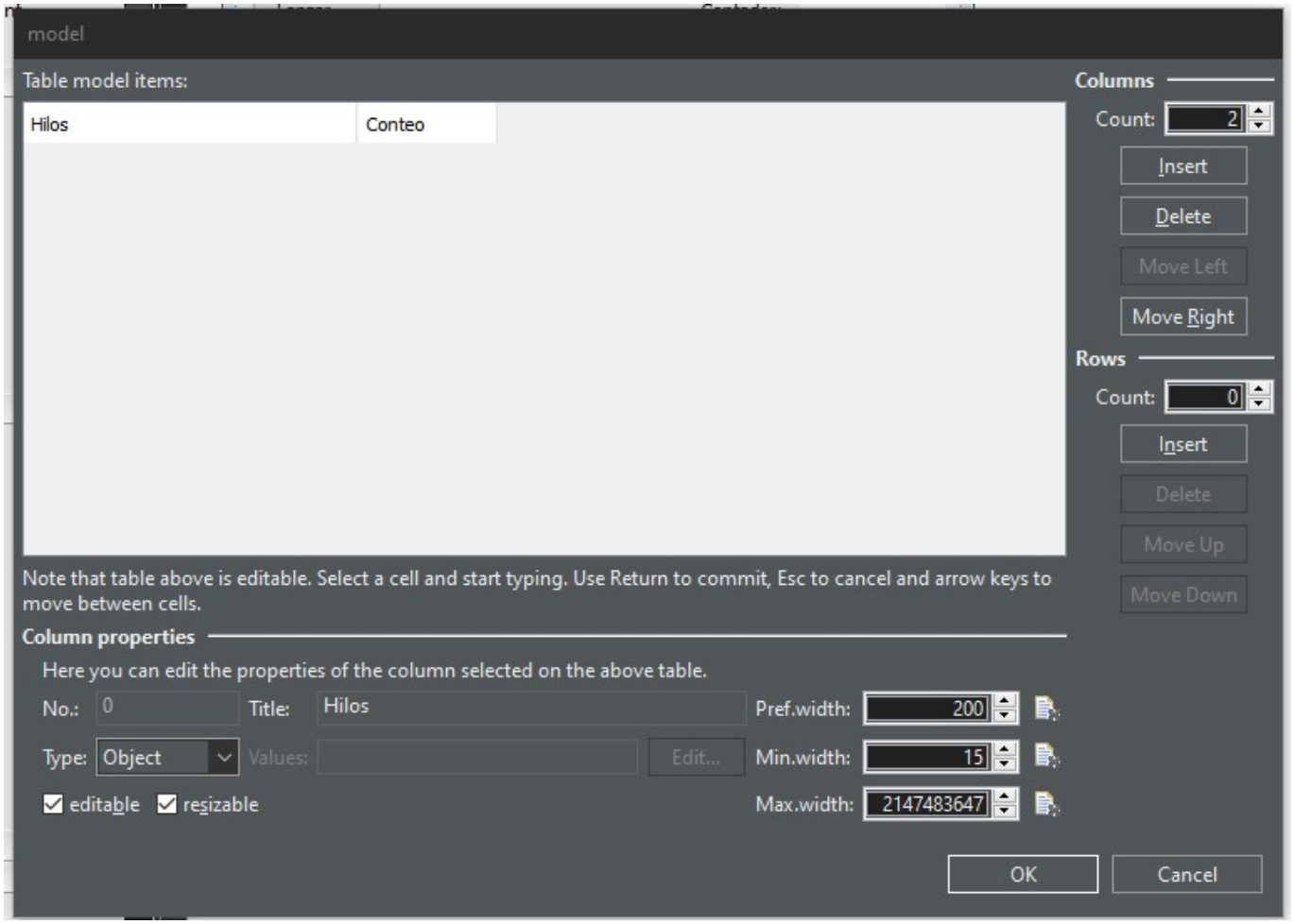
A continuación insertamos la JTable en la ventana y en propiedades de esta vamos a model:



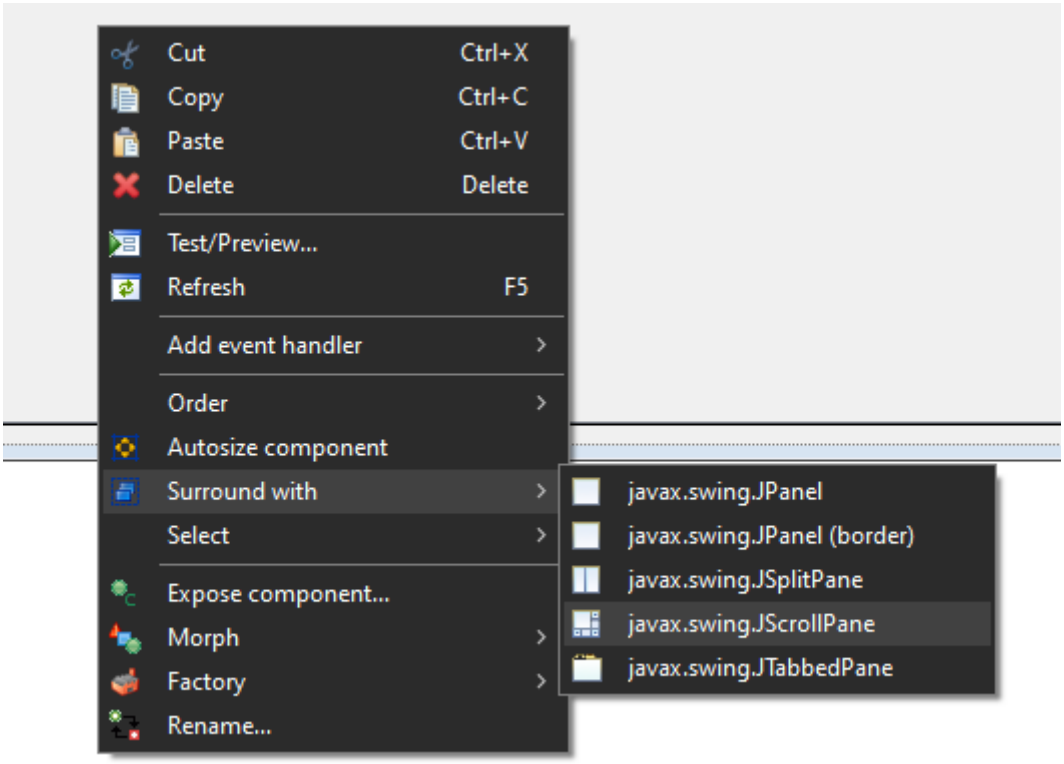
Cuando le demos a model nos saldrá la siguiente ventana con el editor de la tabla:



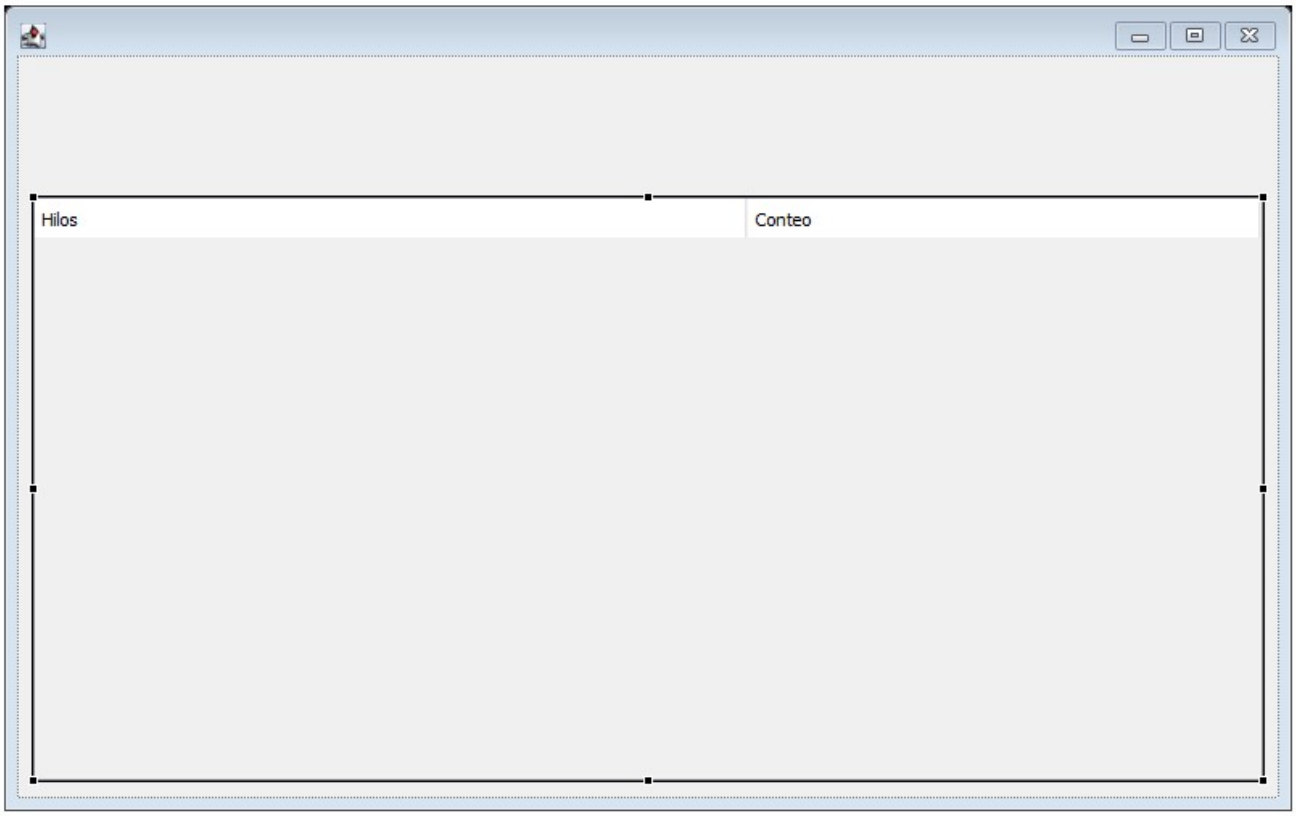
En la derecha le daremos a insertar una vez, colocaremos abajo el nombre de la columna, en este caso conteo, le daremos a insertar de nuevo y pondremos como nombre Hilos. Podemos ajustar el tamaño de las columnas abajo, en este caso la columna Hilos será de tamaño 200:



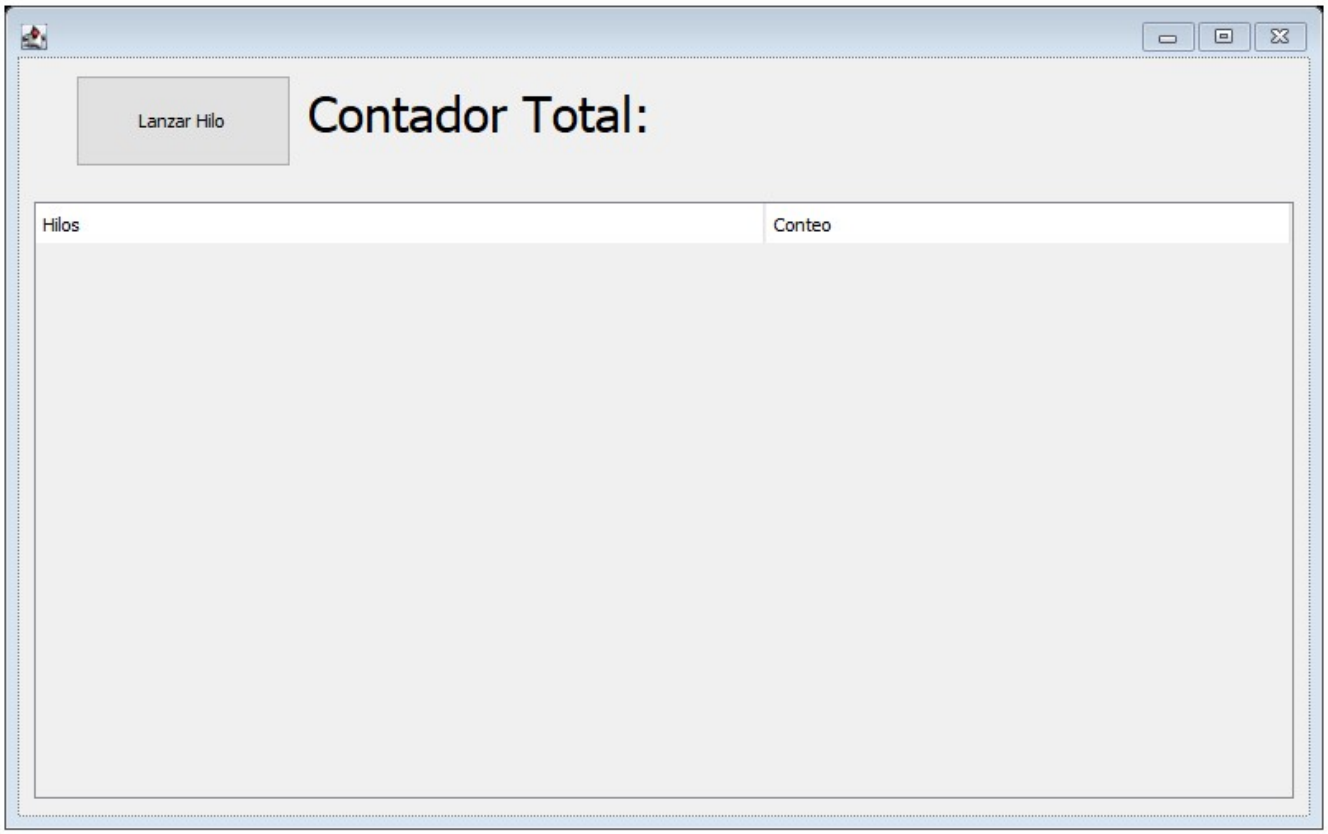
Tras darle a aceptar, no veremos nada en donde hemos insertado la tabla, ahora le daremos click derecho al recuadro donde debería estar la tabla e iremos a Surround With y seleccionaremos JScrollPane:



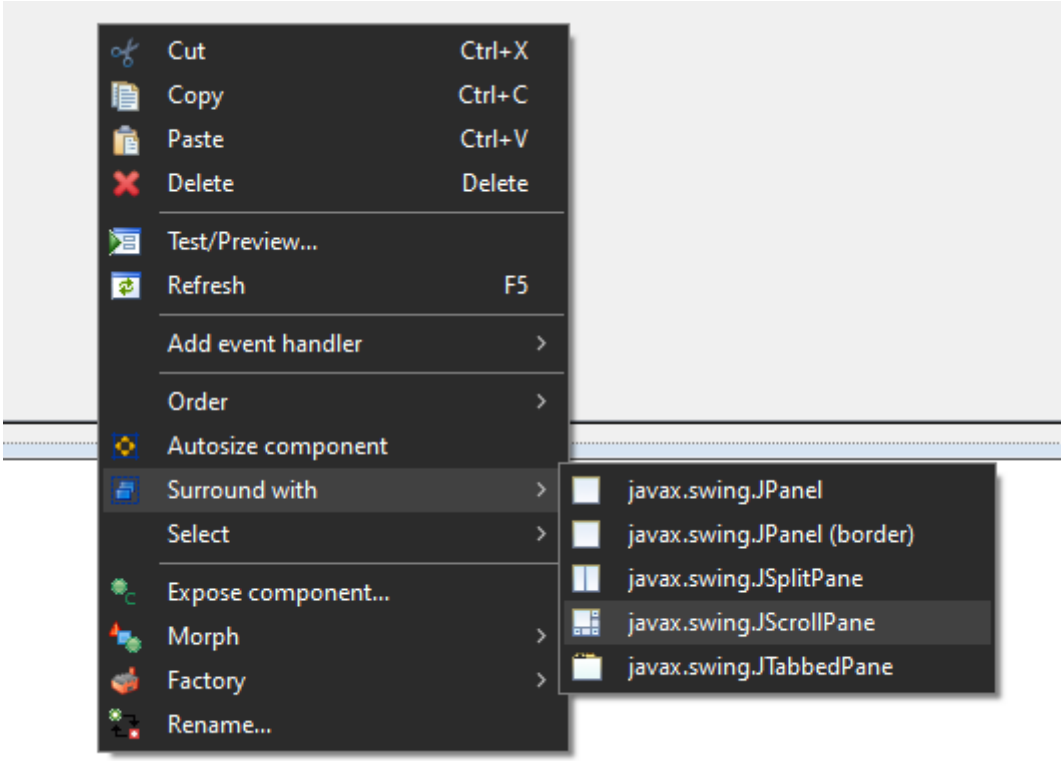
Tras eso tendremos algo como esto:



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:

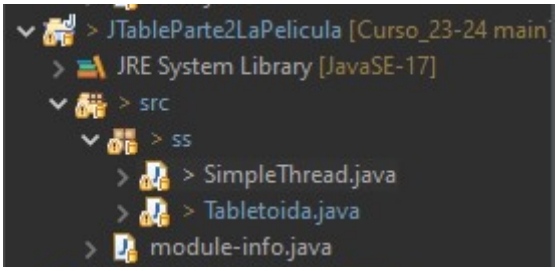


Finalmente insertaremos un JTextField no editable para que muestre los conteos totales:



Implementación Funcional 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 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);
```

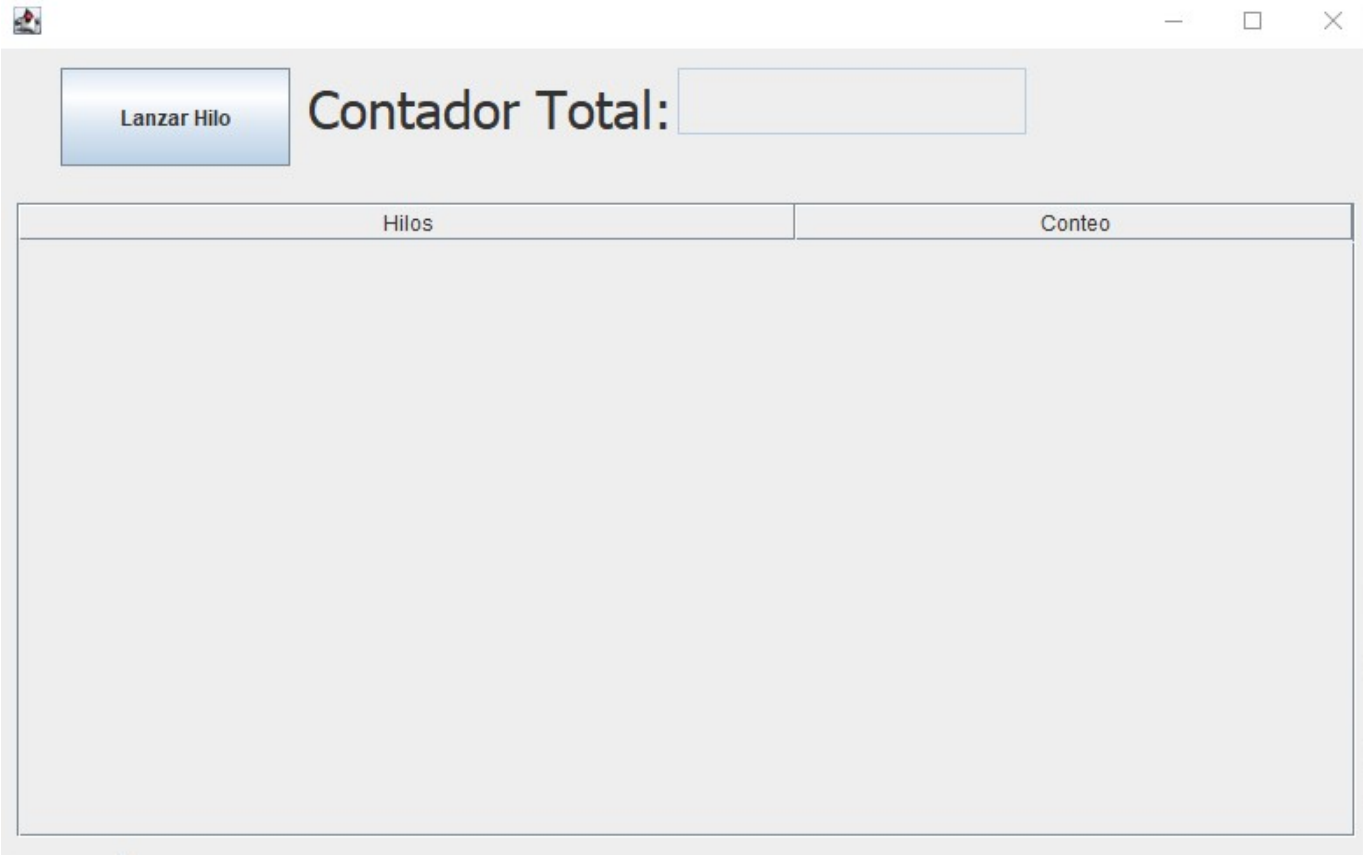
Botón Lanzar

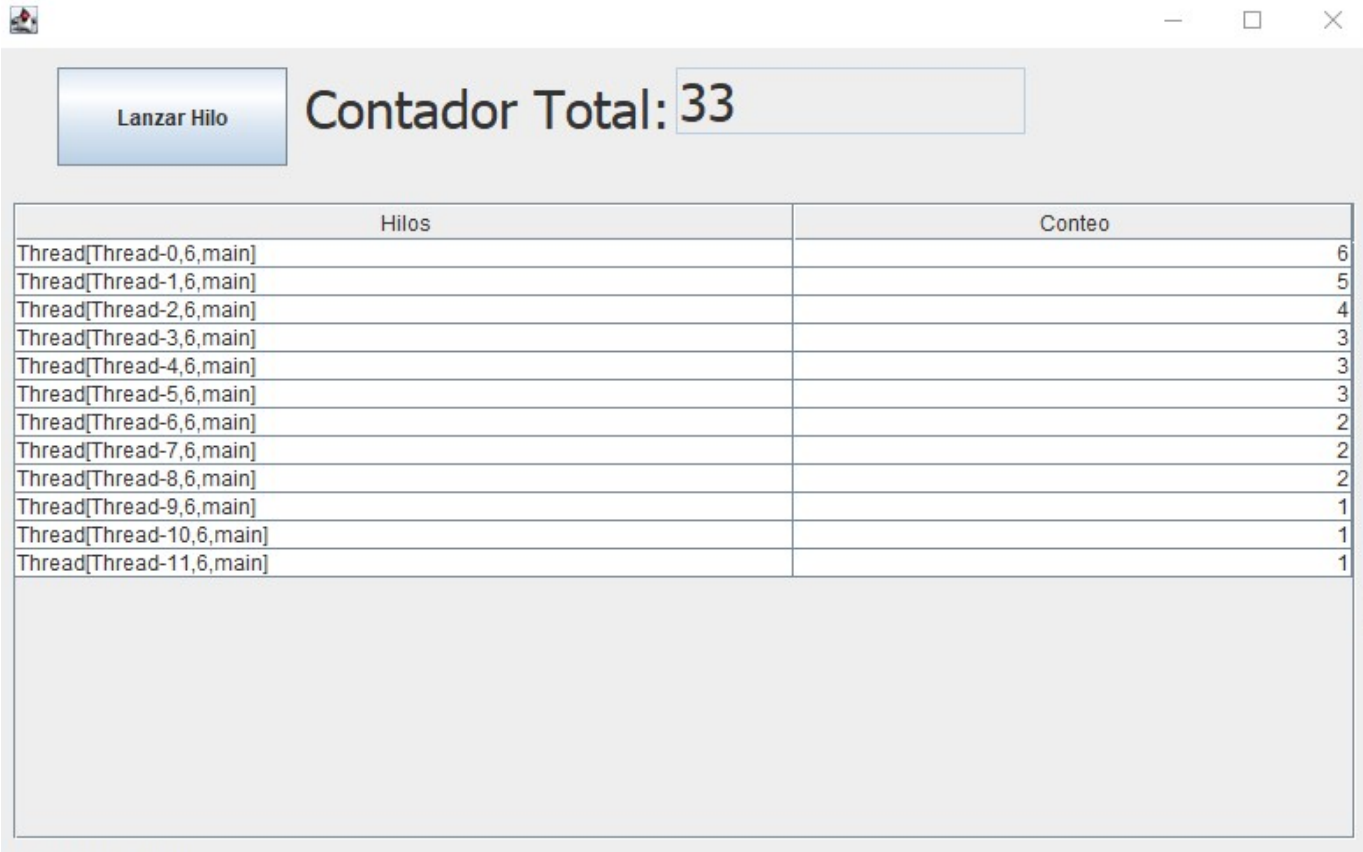
Ahora haremos que cada vez que pulsemos el botón cada hilo que arranquemos se vaya almacenando en una lista de hilos y que tras eso, vuelque esta lista de hilos a la Jtable para Rellenarla. A su vez, se sumará el valor de los campos que almacenen el conteo de iteraciones de cada hilo en una variable 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 pulsemos 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).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);
```

Resultado

Ahora la tabla se verá de la siguiente forma:





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 las lista de de hilos a partir de la fila y columna en la que estemos para rellenar la tabla.

```java
package ss;

import ss.Tabletoida;
import ss.SimpleThread;
import javax.swing.table.DefaultTableModel;

public class ModeloTabla extends DefaultTableModel{
    private Tabletoida ventana = null;//Objeto de tipo Tabletoida, invoca la ventana principal

    public ModeloTabla(Tabletoida ventana) {//constructor del modelo
        this.ventana = ventana;
    }

    @Override
    public int getColumnCount() {
        return this.ventana.getColName().length;//Contamos cuantas columnas tiene la tabla
    }

    @Override
    public String getColumnName(int column) {//Obtenemos el nombre de la columna indicada
        return this.ventana.getColName()[column];
    }

    @Override
    public int getRowCount() {//contamos cuantas filas hay
        int rowCount=0;
        if(this.ventana != null) {
            rowCount = this.ventana.getListaHilos().size();
        }
        return rowCount;
    }

    @Override
    public Object getValueAt(int row, int column) {//Obtener el valor almacenado en la lista que corresponde a la posición indicada
        Object returnValue = null;
        if (this.ventana.getListaHilos().size()>0) { // Si hay hilos insertados en la lista
            SimpleThread hilo = (SimpleThread)this.ventana.getListaHilos().get(row);//declaramos hilo con el valor del que se encuentra en cierta fila "row"
            switch (column) {//En funcion de la columna que queramos obtener (0 para el hilo y 1 para el conteo)
                case 0://Obtenemos ID de Hilo
                    returnValue = hilo.getId();
                    break;
                case 1://Obtenemos Valor del contador del hilo en cuestión
                    returnValue = hilo.getLecontadorInterno();
                    break;
                default://en caso de solicitar otra cosa, no se devuelve nada
                    returnValue = "";
            }
        }
        return returnValue;//Devolvemos el elemento seleccionado.
    }

    public boolean isCellEditable(int row, int column) {//Hacemos que no se puedan editar las celdas de la tabla
          return false;
       }

}

```

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

Permanent link:
https://knoppia.net/doku.php?id=dad:jtable&rev=1695979181

Last update: **2023/09/29 09:19**