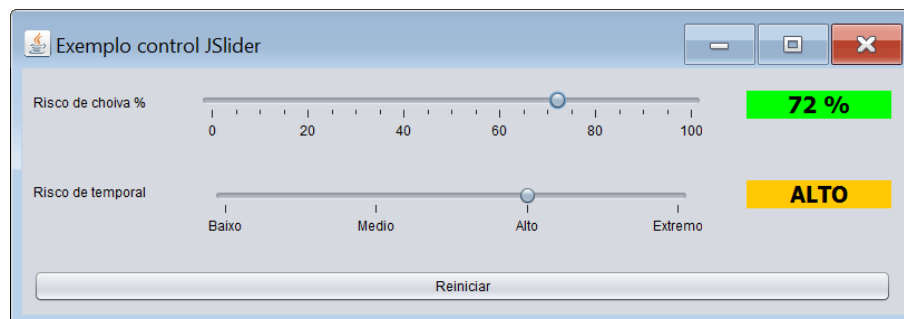


# 1. Componente Deslizador (JSlider)

O deslizador é un compoñente gráfico que almacena un rango de valores predefinidos, permitindo ao usuario seleccionar un deles. Os valores que xestiona son valores enteiros. Os valores que conforman o rango de valores predefinidos represéntanse mediante unha liña, sendo un extremo desa liña o valor mínimo do rango de valores posibles e o outro extremo da liña o valor máximo do rango de valores posibles. O usuario poderá arrastrar un marcador ao longo da liña para indicar de tódolos valores posibles cal é o seleccionado. Este control é substituíble por un espíner e viceversa, sendo moitas veces unha cuestión de espazo e/ou estética o empregar un ou outro. O compoñente deslizador defínese a través da clase `javax.swing.JSlider`. Gráficamente o seu aspecto é o seguinte:



Na imaxe anterior podemos ver o emprego de dous deslizadores para seleccionar valores dentro dun rango de valores posibles.

Propiedades do control JSlider empregadas máis habitualmente e que pódense establecer a través do entorno de programación:

Propiedade	Función
BackGround	Cor de fondo do compoñente
Border	Borde do compoñente
Cursor	Mediante esta propiedade indícase a imaxe que amosará o punteiro do rato ao navegar sobre o compoñente
Enabled	Se esta propiedade está activada o compoñente será funcional. No caso de que esta propiedade estea desactivada o compoñente será visualizable, pero o usuario non poderá interaccionar con el.
Focusable	Se esta propiedade está activada o compoñente entrará na roda de reparto do foco de xeito que ao premer a tecla de cambio de foco (habitualmente o tabulador) nalgún momento tomará o foco da aplicación (cando sexa a súa quenda na roda de reparto do foco). Pola contra, se esta propiedade está desactivada o compoñente non entrará na roda de reparto do foco e soamente será posible acceder a el mediante o rato.
Font	Características da fonte do compoñente (tipo de fonte, tamaño, ... )
Foreground	Cor do texto do compoñente
Inverted	Se esta propiedade está desactivada, os valores do deslizador amosaranse de menor a maior. Se está activada amosaranse de maior a menor.
MajorTicksSpacing	Separación (medida en valores do rango) entre os major ticks do deslizador
Maximum	Valor máximo do rango do deslizador
Minimum	Valor mínimo do rango do deslizador

MinorTicksSpacing	Separación (medida en valores do rango) entre os minor ticks do deslizador
Orientation	Orientación do deslizador. Pode ser horizontal ou vertical.
PaintLabels	Indica se pintar ou non os major ticks. Os major ticks son marcas (de maior tamaño que os minor ticks) no deslizador que se empregan como referencias para saber en que valores estámonos a mover dentro do deslizador. Adxunto a cada major tick amósase o seu valor dentro do rango de valores posibles. Ademais, é posible substituír ese valor por unha etiqueta de texto (esta operación realízase a través de código).
PaintTicks	Indica se pintar ou non os minor ticks. Os minor ticks son pequenas marcas no deslizador que se empregan como referencias para saber en que valores estámonos a mover dentro do deslizador.
PaintTrack	Indica se pintar ou non a liña do deslizador
SnapToTicks	Mediante esta propiedade indicamos que ocorrerá cando soltemos o marcador do deslizador entre dos ticks. Se está activada, o marcador moverase ata o tick máis próximo. Se non está activada, o marcador quedarase onde o solte o usuario.
ToolTipText	Mediante esta propiedade establécese unha mensaxe (tooltip) que será amosado ao deixar o punteiro do rato sobre o compoñente. É habitual empregar esta mensaxe para indicar cal é a utilidade do compoñente
Value	Valor inicial do deslizador. Ten que ser un valor dentro do rango posible do compoñente.

Eventos do control JSlider empregados máis habitualmente e que pódense establecer a través do entorno de programación:

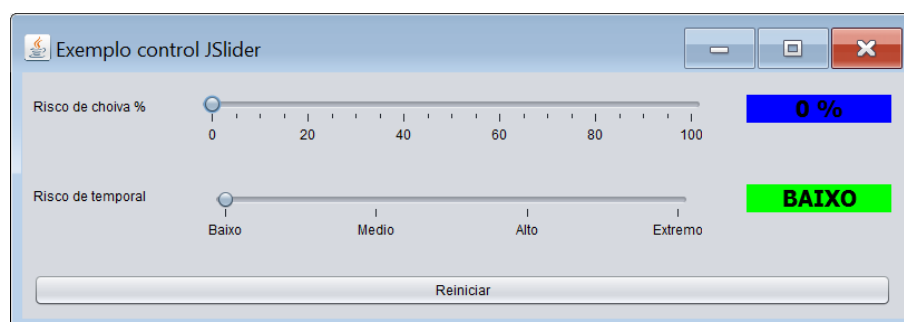
Evento	Lanzamento
StateChanged	Este evento é disparado cando cambiamos el elemento seleccionado no deslizador.

Métodos do control JSlider empregados máis habitualmente:

Método	Función
Public void setValue(int n)	Establece o elemento seleccionado no deslizador. Debe ser un elemento entre os valores posibles que pode tomar o compoñente.
Public int getValue()	Devolve o elemento seleccionado no deslizador.

## Xestión de deslizadores empregando NetBeans

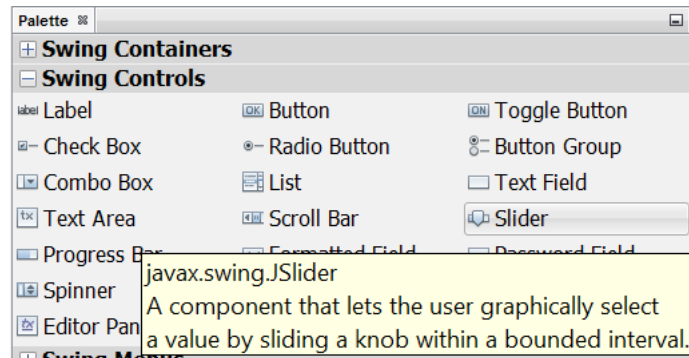
A continuación imos desenvolver o seguinte formulario:



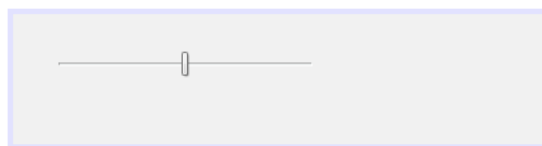
O formulario consta de dous deslizadores. O primeiro deslizador emprégase para indicar o risco de choiva en porcentaxe. En función do valor seleccionado no deslizador amósase na etiqueta adxunta o valor dese porcentaxe. Ademais a etiqueta adxunta cambiará de color en función do valor do porcentaxe. O segundo deslizador emprégase para indicar o risco de temporal. En función do valor seleccionado no deslizador amosarase na etiqueta adxunta o risco de

temporal. Ademais a etiqueta adxunta cambiará de color en función do risco seleccionado. O botón Reiniciar reseteará o formulario ao seu estado inicial. Calquera erro que poida acontecer ao longo do emprego do programa debe ser reportado ao usuario.

Para engadir un compoñente de tipo JSlider no noso formulario primeiramente seleccionáremolo da paleta de compoñentes do entorno de programación:

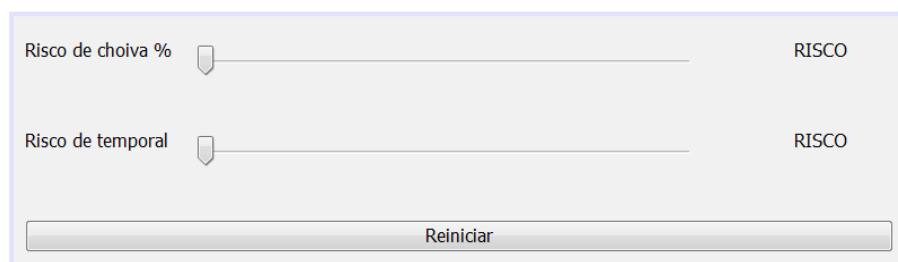


e arrastrámolo ata o formulario:

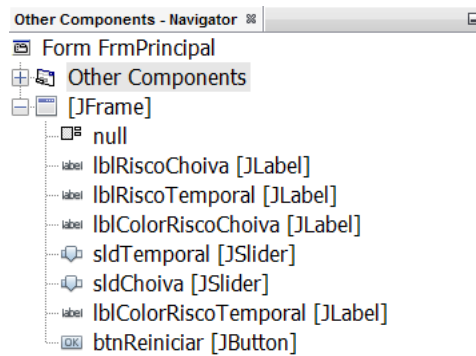


Unha vez situado o compoñente dentro do formulario, adaptámolo para que teña o aspecto visual que desexamos que teña.

No caso que estamos desenvolvendo necesitamos engadir catro etiquetas, un botón e dous deslizadores. Unha vez engadidos e colocados no seu lugar correspondente, este será o resultado do deseño:



O noso formulario quedará configurado do seguinte xeito:



Unha vez que temos colocados os compoñentes que necesitamos hai que configurar o seu aspecto. Neste caso faremos as seguintes modificacións:

- Para as etiquetas `lblColorRiscoChoiva` e `lblColorRiscoTemporal`, modificaremos a súa propiedade `font` para que a letra sexa negra e tamaño vinte. Ademais o texto estará centrado respecto á etiqueta.
- Para o deslizador `sldChoiva` modificaremos varias propiedades. Primeiro, como imos representar un porcentaxe, o seu rango de valores posibles estará entre cero e cen. Para indicar isto dámoslle á súa propiedade `minimum` o valor 0 e á súa propiedade `maximum` o valor 100. Á súa propiedade `value` dámoslle 0 para que comence nesa posición. Respecto aos ticks, queremos que se amosen os pequenos e os grandes. Para os pequenos, dámoslle á súa propiedade `minorTickSpacing` o valor 5 indicando que teremos unha separación de 5 valores do rango entre eles e ademais marcamos a propiedade `paintTicks` para que os pinte. Respecto aos ticks grandes, dámoslle á súa propiedade `majorTickSpacing` o valor 20 indicando que teremos unha separación de 20 valores do rango entre eles e ademais marcamos a propiedade `paintLabels` para que os pinte.
- Respecto ao deslizador `sldTemporal`, modificaremos varias propiedades. Primeiro, como imos representar catro valores dunha escala (baixo, medio, alto e extremo), o seu rango de valores posibles estará comprendido entre 0 e 3. Para indicar isto dámoslle á súa propiedade `minimum` o valor 0 e á súa propiedade `maximum` o valor 3. Á súa propiedade `value` dámoslle 0 para que comence nesa posición. Respecto aos ticks, só queremos que se amosen os grandes, polo tanto á súa propiedade `majorTickSpacing` dámoslle o valor 1 indicando que haberá unha separación dun valor do rango entre eles e ademais marcamos a propiedade `paintLabels` para que os pinte (a asignación de etiquetas aos ticks grandes será feita mediante código).

O comportamento visual da aplicación practicamente está resolto (unicamente faltaríanos xestionar o cambio de textos e cores das etiquetas en función das accións do usuario sobre o formulario). Agora imos desenvolver a súa funcionalidade.

Primeiramente imos ver como asignar as etiquetas de texto que se amosaran no deslizador `sldTemporal` para substituír aos ticks grandes que proporciona por defecto o sistema. Para realizar esta operación escribimos o seguinte código:

```
public FrmPrincipal() {
    initComponents();
    Hashtable etiquetasRiscoTemporal = new Hashtable();
    etiquetasRiscoTemporal.put(new Integer(0), new JLabel("Baixo"));
    etiquetasRiscoTemporal.put(new Integer(1), new JLabel("Medio"));
    etiquetasRiscoTemporal.put(new Integer(2), new JLabel("Alto"));
}
```

```
etiquetasRiscoTemporal.put( new Integer(3), new JLabel("Extremo"));
this.sldTemporal.setLabelTable(etiquetasRiscoTemporal);
```

O mellor sitio para facelo é no construtor do formulario, despois da chamada a  `initComponents` . Como pódese observar, o primeiro que facemos é crear un  `Hashtable`  cuxos índices son os valores dos ticks grandes que queremos substituír e os contidos das posicións indicadas polos índices son compoñentes  `JLabel`  con os textos que queremos que se amosen en lugar dos ticks grandes amosados por defecto:

```
Hashtable etiquetasRiscoTemporal = new Hashtable();
etiquetasRiscoTemporal.put(new Integer(0), new JLabel("Baixo"));
etiquetasRiscoTemporal.put( new Integer(1), new JLabel("Medio"));
etiquetasRiscoTemporal.put( new Integer(2), new JLabel("Alto"));
etiquetasRiscoTemporal.put( new Integer(3), new JLabel("Extremo"));
```

A continuación temos que ligar a  `Hashtable`  creada co deslizador. Para elo emprégase o método  `setLabelTable`  sobre o deslizador  `sldTemporal` :

```
this.sldTemporal.setLabelTable(etiquetasRiscoTemporal);
```

A partir deste momento amosaranse como ticks grandes do deslizador  `sldTemporal`  as etiquetas definidas na  `Hashtable` .

Os requirimentos do programa pídenos que cando modifiquemos o valor del deslizador  `sldChoiva`  sexa amosado na etiqueta adxunta o valor dese porcentaxe e que ademais a etiqueta adxunta cambie de cor en función do valor do porcentaxe. Cando realizamos algún cambio nos valores dun deslizador dispárase o seu evento  `stateChanged` . Polo tanto, a fin de cumprir os requirimentos de noso programa debemos implementar o evento  `stateChanged`  do deslizador  `sldChoiva` . Vexamos a continuación a súa implementación:

```
private void sldChoivaStateChanged(javax.swing.event.ChangeEvent evt) {
    // TODO add your handling code here:
    if(!sldChoiva.getValueIsAdjusting())
    {
        int valorSlider=sldChoiva.getValue();
        lblColorRiscoChoiva.setText(new Integer(valorSlider).toString()+" %");
        if(valorSlider<20)
        {
            lblColorRiscoChoiva.setBackground(Color.blue);
        }
        else
        {
            if(valorSlider>=20 && valorSlider<40)
            {
                lblColorRiscoChoiva.setBackground(Color.green);
            }
            else
            {
                if(valorSlider>=40 && valorSlider<60)
                {
                    lblColorRiscoChoiva.setBackground(Color.yellow);
                }
                else
                {
                    if(valorSlider>=60 && valorSlider<80)
                    {
                        lblColorRiscoChoiva.setBackground(Color.green);
                    }
                    else
                    {
                        lblColorRiscoChoiva.setBackground(Color.red);
                    }
                }
            }
        }
    }
}
```

Cando desprazamos o marcador dun deslizador sin soltalo, cada vez que pasa por un novo valor dispárase o evento  `stateChanged` . No noso caso unicamente estamos interesados en capturar o valor do deslizador unha vez que se solta o seu marcador. Para determinar se estamos desprazando o marcador do deslizador o se foi soltado empregase o método  `getValueIsAdjusting`  sobre o deslizador. Este método devolveranos  `true`  se non foi soltado e  `false`  no caso de haber liberado o marcador do deslizador:

```
if (!sldChoiva.getValueIsAdjusting())
```

Unha vez que o marcador do slider foi liberado unicamente temos que recuperar o seu valor e en función del establecer un texto determinado e unha cor determinada para a etiqueta asociada ao deslizador sldChoiva:

```
int valorSlider=sldChoiva.getValue();  
lblColorRiscoChoiva.setText(new Integer(valorSlider).toString()+" %");  
if(valorSlider<20)  
{  
    lblColorRiscoChoiva.setBackground(Color.blue);  
}
```

Para recuperar o valor seleccionado do deslizador empregamos o método `getValue` sobre o deslizador `sldChoiva`.

O tratamento do deslizador `sldTemporal` é similar. A continuación achégase o código implementado no seu evento `StateChanged`:

```
private void sldTemporalStateChanged(javax.swing.event.ChangeEvent evt) {  
    // TODO add your handling code here:  
    if (!sldTemporal.getValueIsAdjusting())  
    {  
        switch(sldTemporal.getValue())  
        {  
            case 0: lblColorRiscoTemporal.setText("BAIXO");  
                    lblColorRiscoTemporal.setBackground(Color.green);  
                    break;  
            case 1: lblColorRiscoTemporal.setText("MEDIO");  
                    lblColorRiscoTemporal.setBackground(Color.yellow);  
                    break;  
            case 2: lblColorRiscoTemporal.setText("ALTO");  
                    lblColorRiscoTemporal.setBackground(Color.orange);  
                    break;  
            case 3: lblColorRiscoTemporal.setText("EXTREMO");  
                    lblColorRiscoTemporal.setBackground(Color.red);  
                    break;  
        }  
    }  
}
```

Por último, os requirimentos do programa indícanos que ao pulsar o botón Reiniciar resetearase o formulario ao seu estado inicial. En relación cos deslizadores isto implica asignarlles o valor que teñan ao arrancar a aplicación. Iremos ver como realizar esta asignación para o deslizador `sldChoiva`:

```
sldChoiva.setValue(0);
```


Simplemente temos que empregar o método `setValue` do deslizador e pasarlle o valor que queremos darlle. Este valor deberá estar comprendido dentro do rango de valores posibles que pode tomar o deslizador.

Respecto á asignación dun valor entre os admisibles para o deslizador `sldTemporal`, empregamos o seguinte código:

```
sldTemporal.setValue(0);
```

Ao igual que fixemos co deslizador `sldChoiva`, facendo emprego do método `setValue` do deslizador, pasámoslle o valor dun dos elementos admisibles polo deslizador.

Na seguinte imaxe amósase a execución da aplicación:

 Exemplo control JSlider

Risco de chuva %

020406080100

82 %

Risco de temporal

BaixoMedioAltoExtremo

ALTO

Reiniciar