1. Xestión doutros compoñentes

Cando desenvolvemos aplicacións gráficas de usuario cada un dos formularios que compoñen as nosas aplicacións están compostos por unha serie de compoñentes gráficos que son empregados para levar a cabo a comunicación de información entre o usuario e a aplicación. Cos coñecementos acadados sobre compoñentes básicos e avanzados estamos en disposición de enfrontarnos ao desenvolvemento de practicamente calquera formulario. Non obstante, existe unha grande diversidade de compoñentes gráficos ademais dos que acabamos de ver. Co fin de estender un pouco máis o noso coñecemento sobre compoñentes gráficos imos estudar outros compoñentes gráficos máis que nos ofrece o entorno de programación NetBeans. Algúns destes compoñentes empréganse con certa regularidade, mentres que outros empréganse en ocasións excepcionais. A pesar de non ser compoñentes de emprego moi habitual sempre é interesante coñecelos xa que nos poden aforrar moito traballo de desenvolvemento. Do mesmo xeito que fixemos cos compoñentes gráficos vistos ata o momento, centrarémonos nas tarefas para as que se soen empregar estes outros compoñentes máis habitualmente, xa que estudar calquera destes compoñentes en profundidade sería un traballo por un lado excesivo, debido a que son moitísimas as propiedades, eventos e métodos que xestiona cada un deles, e por outra banda sería unha perda de tempo, xa que estudaríamos funcionalidades que rara vez vanse empregar. É mellor idea saber xestionar cada compoñente para empregar as súas funcionalidades habituais e no caso de ter que empregar algunha funcionalidade excepcional, aprender como xestionar esa funcionalidade excepcional en concreto. Para coñecer tódalas propiedades, eventos e métodos de cada un destes outros compoñentes sempre poderemos acceder á API (application programming interface) de cada compoñente. De seguido imos enumerar os diferentes compoñentes que imos estudar:

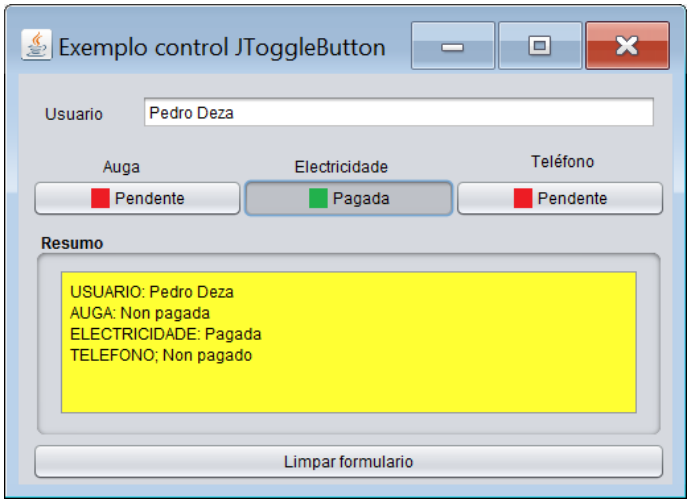
- Botón de panca
- Espíner
- Deslizador
- Barra de progreso
- Caixa de texto para contrasinais

2. Compoñente Botón de panca (JToggleButton)

O compoñente JToggleButton, a pesar de que ten a aparencia dun botón, compórtase dun xeito moi similar ás casillas de verificación e aos botóns de opción. Un botón de panca, ao igual que unha casilla de verificación ou un botón de opción pode ser marcado ou desmarcado polo usuario, sendo tarefa do programador o asignar eses estados de marcado ou desmarcado á asignación lóxica interna do programa. Habitualmente un botón de panca vai acompañado dun texto (provisto polo propio compoñente) que é empregado para indicar ao usuario do interface gráfico cal é a utilidade do compoñente. Ademais de ser empregado para tomar datos de entrada, tamén pode usarse para amosar o estado dalgún elemento durante a execución do noso programa. Tipicamente nos formularios nos que hai botóns de panca existe algún control (normalmente un

botón) sobre o cal o usuario pode actuar de xeito que esa actuación desencadee a recollida dos estados dos botóns de panca para o seu posterior procesamento. En moitas ocasións o actuar sobre un botón de panca implica o desencadeamento de eventos que cambian o aspecto da interface de usuario ou modifican a funcionalidade do programa. Tamén cabe mencionar que podemos cambiar o aspecto do control para que no botón de panca amósense diferentes imaxes (acompañadas ou non de texto) en función do estado do botón de panca.

O compoñente botón de panca defínese a través da clase javax.swing.JToggleButton. Graficamente o seu aspecto é o seguinte:



Na imaxe anterior podemos ver 3 botóns de panca acompañados por outros controis.

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

Propiedade	Función
Background	Cor de fondo do compoñente
ButtonGroup	Emprégase para indicar a que grupo de botóns de panca pertence este botón de panca. Un grupo de botóns de panca permite que unicamente un dos botóns de panca dos pertencentes ao grupo poda estar activo.
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
HorizontalTextPosition	Localización horizontal do texto do compoñente respecto á imaxe amosada polo compoñente
Icon	Mediante esta propiedade indícase a imaxe a amosar no compoñente. Se non indicamos nada o compoñente amosará o seu aspecto estándar.
IconTextGap	Separación entre o texto do compoñente e a imaxe empregada polo compoñente
PressedIcon	Mediante esta propiedade indícase a imaxe a amosar cando prememos sobre o compoñente.

RollOverEnabled	Se esta propiedade está desactivada, cando pasemos sobre o compoñente amosarase a imaxe indicada na súa propiedade RollOverIcon. Se RollOverIcon non está definida non se amosará ningunha imaxe. Pola contra, se a propiedade RollOverEnabled está activada, cando pasemos sobre o compoñente amosarase a imaxe indicada na súa propiedade RollOverIcon. Se RollOverIcon non está definida amosarase a imaxe definida na súa propiedade SelectedIcon e se a propiedade SelectedIcon non está definida non se amosará nada.
RollOverIcon	Mediante esta propiedade indícase a imaxe a amosar cando pasamos co rato sobre o compoñente.
Selected	Indica se o botón de panca está marcado ou non
SelectedIcon	Mediante esta propiedade indícase a imaxe a amosar cando o compoñente está seleccionado.
Text	Emprégase para establecer o texto que aparece no botón de panca
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
VerticalTextPosition	Localización vertical do texto do compoñente respecto á imaxe amosada polo compoñente

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

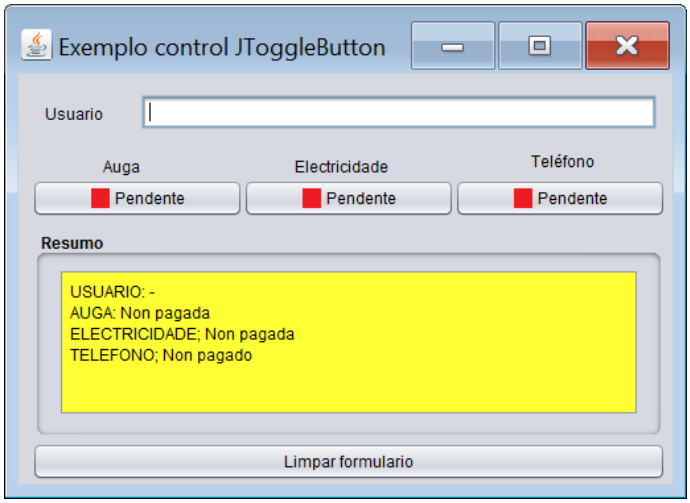
Evento	**Lanzamento**
ItemStateChanged	Este evento é disparado cando cambiamos o estado do botón de panca, xa sexa por acción directa do usuario ou a través de código.

Métodos do control JToggleButton empregados máis habitualmente:

Método	**Función**
public boolean isSelected()	Emprégase para recuperar o estado do botón de panca
public void setSelected(boolean b)	Emprégase para establecer o estado do botón de panca

Xestión de botóns de panca empregando NetBeans

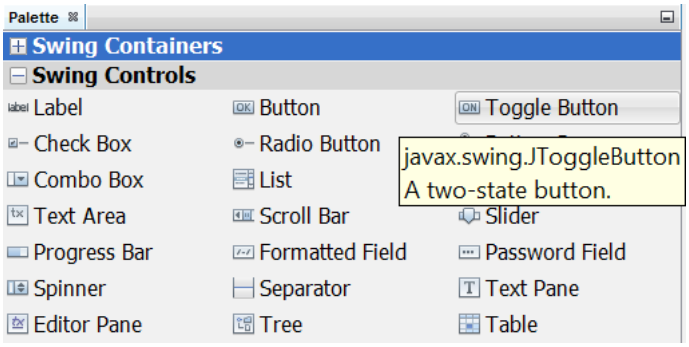
A continuación imos desenvolver o seguinte formulario:



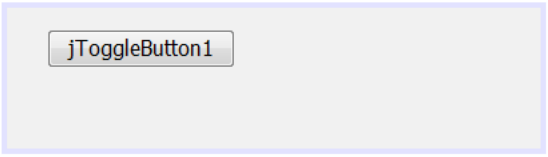
O formulario consta de tres botóns de panca que empregamos para indicar se o usuario ten ou non pagados diferentes servizos. Ao pulsar sobre un botón de panca, este modificará a imaxe e o texto do propio botón de panca (de Pendente a Pagado e viceversa) e actualizarase a información dispoñible sobre o usuario

na área de texto Resumo indicando o seu novo estado. Ao pulsar sobre o botón Limpar formulario volverase ao estado inicial do formulario. Ademais calquera pulsación sobre a caixa de texto tamén desencadeará a actualización da información referida na área de texto Resumo. No caso de que ocorra algún erro durante a execución informarase ao usuario para que o emende.

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

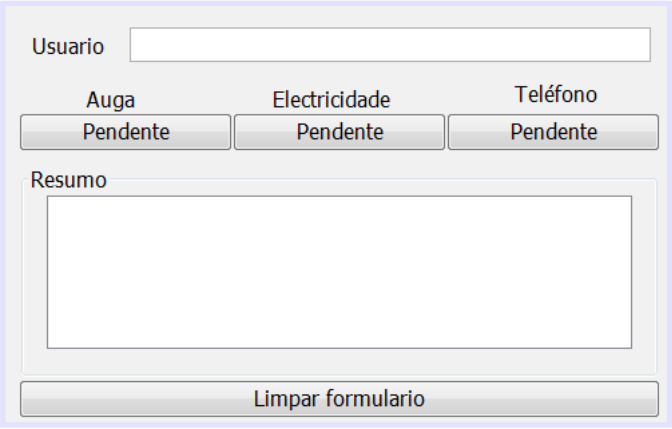


e arrastrámolo ata o formulario:

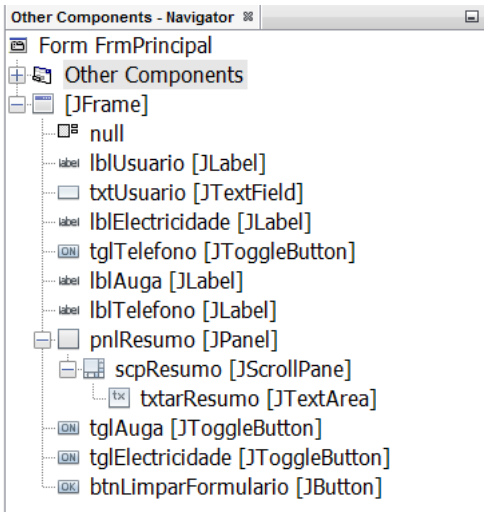


Unha vez situado dentro do formulario, e como sempre, adaptámolo para que teña o aspecto visual que desexamos que teña.

No caso que estamos desenvolvendo necesitamos engadir catro etiquetas, unha caixa de texto, un botón, un panel con borde titulado, un área de texto e tres botóns de panca. Unha vez engadidos e colocados no seu lugar correspondente, este será o resultado do deseño:



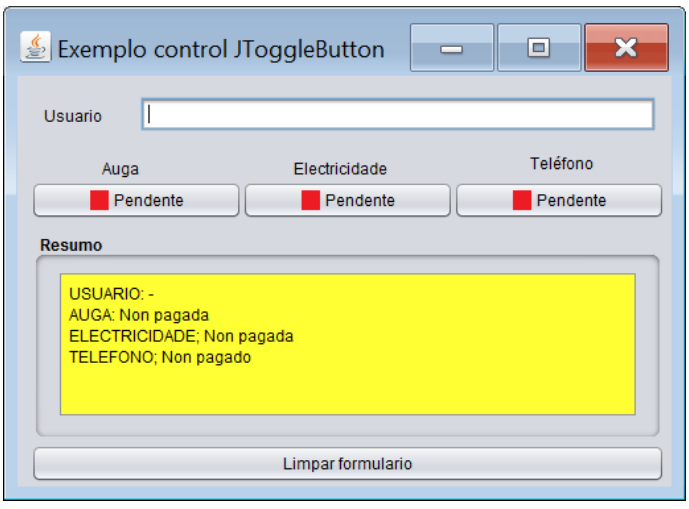
O noso formulario quedará configurado do seguinte xeito:



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

- Para a área de texto txtarResumo, desmarcamos a súa propiedade editable e dámoslle como cor de fondo o amarelo.
- Respecto aos tres botóns de panca, imos modificar as súas propiedades para poder realizar a asignación de imaxes dos mesmos. Á propiedade icon asignámoslle a imaxe que queremos que se amose cando o botón atópese en estado deseleccionado. Á propiedade selectedIcon asignámoslle a imaxe que queremos que se amose cando o botón atópase en estado seleccionado. Ademais desactivamos a propiedade rollOverEnabled para evitar o cambio de imaxes ao pasar o rato sobre o botón de panca.

Se executamos a aplicación este será o resultado:



Como pódese observar o comportamento visual da aplicación xa está resolto. Agora imos desenvolver a súa funcionalidade.

Os requirimentos do programa indícannos que cando cambiemos o estado dun botón de panca débese modificar o texto e a imaxe do botón de panca e reflectirase ese cambio nas descricións asociadas ao botón de panca na área de texto. Cando cambia o estado dun botón de panca dispárase o seu evento

ItemStateChanged. A continuación amósase o código asociado a este evento para o botón de panca tglAuga:

```
    private void tglAugaItemStateChanged(java.awt.event.ItemEvent evt) {
        // TODO add your handling code here:
        if(evt.getStateChange()==ItemEvent.SELECTED)
        {
            tglAuga.setText("Pagada");
        }
        else
        {
            tglAuga.setText("Pendente");
        }
        txtarResumo.setText(calcularResumo());
    }
```

Cando o evento ItemStateChanged é disparado, para comprobar se o novo estado do botón de panca é seleccionado ou deseleccionado empregamos a información que nos aporta o evento a través do seu parámetro evt aplicando o método getStateChange. Devolveranos ItemEvent.SELECTED se o botón de panca foi seleccionado e ItemEvent.DESELECTED se foi deseleccionado. En función de se foi seleccionado ou non modificamos o seu texto e por último chamamos ao método calcularResumo que se encargará de construír a cadea de texto que se debe de amosar na área de texto txtarResumo, contendo a información actualizada do estado da aplicación. Opcionalmente, para comprobar se o botón de panca foi seleccionado o deseleccionado tamén poderíamos comprobar o seu estado mediante o seu método isSelected e actuar en consecuencia. Para o resto de botóns de panca que forman parte da nosa aplicación o código é similar.

Respecto ao botón btnLimparFormulario, o cal encárgase de reinicializar a aplicación asignando os valores iniciais da mesma, no tocante ao tratamento dos botóns de panca, o único que hai que indicar é que o seu estado sexa de botón de panca deseleccionado. Para levar a cabo esta acción empregamos o seguinte código:

```
        tglAuga.setSelected(false);
        tglElectricidade.setSelected(false);
        tglTelefono.setSelected(false);
```

Empregando o método setSelected sobre cada un dos botóns de panca, asignámoslles o valor que queremos que tomen, neste caso o valor de deseleccionados.

Na seguinte imaxe pódese observar o funcionamento da aplicación durante a súa execución: