

# 1. Compoñente Casilla de verificación (JCheckBox)

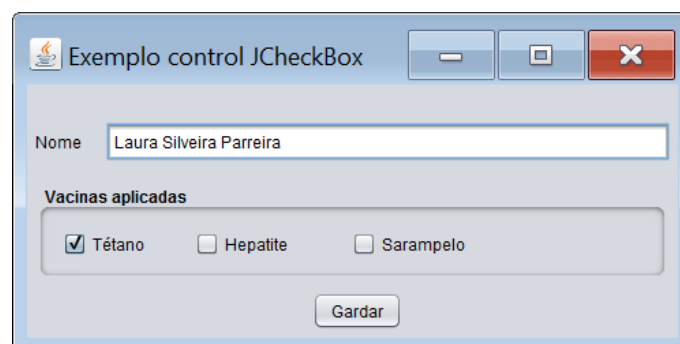
---

Cando desenvolvemos un programa, habitualmente temos que solicitar ao usuario que nos proporcione unha serie de datos de entrada para poder procesalos e xerar un resultado. Cando os datos de entrada que queremos recoller son valores que unicamente poden ter un valor de dous posibles e eses dous valores posibles son si/non ou verdadeiro/falso, nese caso o control máis axeitado é o compoñente de tipo casilla de verificación. Poderíase empregar outro compoñente como por exemplo unha caixa de texto, pero sin lugar a dúbidas fai que a interface sexa menos clara, complica o seu manexo de cara ao usuario e fai a súa programación máis complexa. Unha casilla de verificación é unha pequena caixa que o usuario pode marcar ou desmarcar, sendo tarefa do programador o asignar eses estados de marcado ou desmarcado á asignación lóxica interna do programa. Habitualmente unha casilla de verificación vai acompañada dun texto (provisto polo propio compoñente) que se emprega para indicar ao usuario da interface gráfica cal é a utilidade do compoñente. Ademais de empregarse para tomar datos de entrada, tamén pódese empregar para amosar o estado dalgún elemento durante a execución do noso programa.

Tipicamente nos formularios nos cales hai casillas de verificación existe algún control (normalmente un botón) sobre o cal o usuario pode actuar de xeito que esa actuación provoque a recollida dos datos das casillas de verificación para o seu posterior procesamento. En moitas ocasións o actuar sobre unha casilla de verificación implica o desencadeamento de eventos que cambian o aspecto da interface de usuario (p.e. Ao marcar a casilla de verificación Facturar nun formulario de viaxes habilitase a caixa de texto na cal pódese indicar o número de maletas, mentres que se é desmarcada a casilla de verificación deshabilitase a caixa de texto).

Tamén cabe mencionar que podemos cambiar o aspecto do control para que en lugar de amosarse unha caixa que pode estar ou non marcada amósesse diferentes imaxes (acompañadas ou non de texto) en función do estado da casilla de verificación.

O compoñente casilla de verificación defínese a través da clase `javax.swing.JCheckBox`. Gráficamente o seu aspecto é o seguinte:



Na imaxe anterior podemos ver 3 casillas de verificación acompañadas por outros controis.

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

Propiedade	Función
Background	Cor de fondo 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 premir 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 a casilla de verificación está marcada ou non
SelectedIcon	Mediante esta propiedade indícase a imaxe a amosar cando o compoñente está seleccionado.
Text	Empregase para establecer o texto que aparece na casilla de verificación
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 JCheckBox empregados máis habitualmente e que pódense establecer a través do entorno de programación:

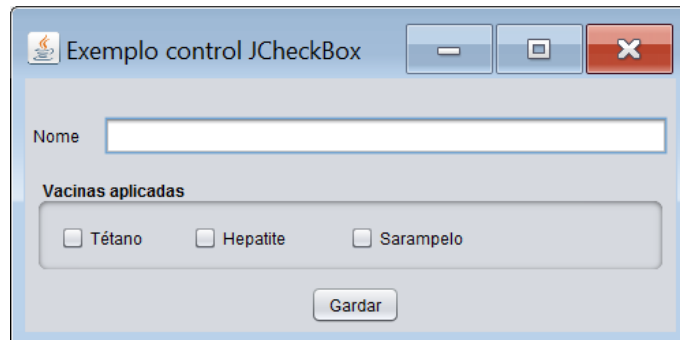
Evento	Lanzamento
ItemStateChanged	Este evento é disparado cando cambiamos o estado da casilla de verificación, xa sexa por acción directa do usuario ou a través de código.

Métodos do control JCheckBox empregados máis habitualmente:

Método	Función
public boolean isSelected()	Empregase para recuperar o estado da casilla de verificación
public void setSelected(boolean b)	Empregase para establecer el valor da casilla de verificación

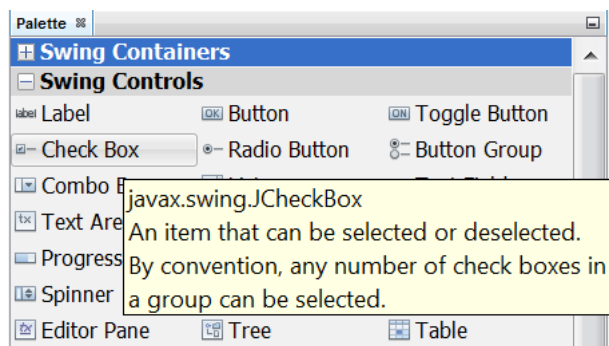
## Xestión de casillas de verificación empregando NetBeans (primeiro exemplo)

A continuación imos desenvolver o seguinte formulario:

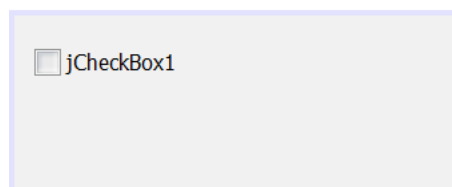


O formulario consta de tres casillas de verificación que empregaremos para indicar as diferentes vacinas aplicadas ou non a unha persoa. Ao pulsar sobre o botón gardar, recolleranse os datos do formulario, validarase a súa corrección e crearase un obxecto que os mapee. Finalmente este obxecto será lanzado a unha base de datos para gardar a súa información. No caso de ocorrer algún erro durante a execución informárase ao usuario.

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



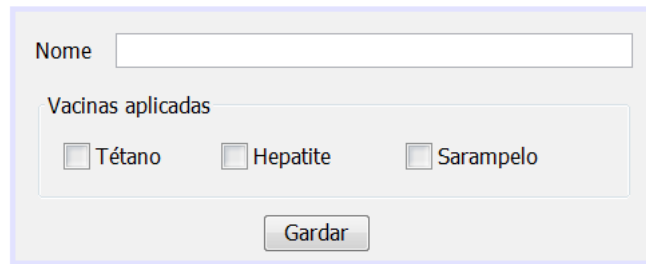
e arrastrámolo ata o formulario:



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

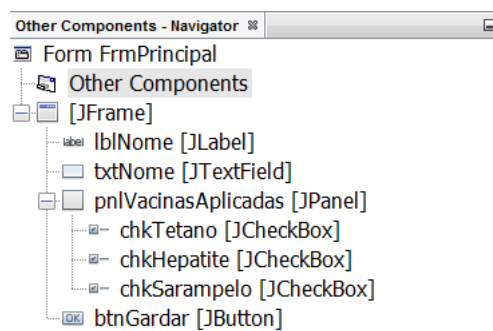
No caso que estamos desenvolvendo necesitamos engadir unha etiqueta, unha caixa de texto, un botón, un panel con borde titulado e tres casillas de

verificación. Unha vez engadidos e colocados no seu lugar correspondente, este será o resultado do deseño:

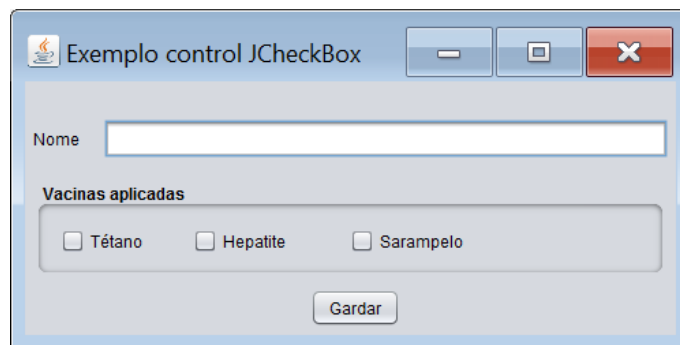


The image shows a visual representation of a Java Swing form design. It features a text field labeled 'Nome' at the top. Below it is a section titled 'Vacinas aplicadas' containing three checkboxes labeled 'Tétano', 'Hepatite', and 'Sarampelo'. At the bottom of the form is a button labeled 'Gardar'.

O noso formulario quedará configurado do seguinte xeito:



Se executamos a aplicación este será o resultado:



The image is a screenshot of the application window titled 'Exemplo control JCheckBox'. It displays the form designed in the previous images, with a text field for 'Nome', a section for 'Vacinas aplicadas' with three checkboxes ('Tétano', 'Hepatite', 'Sarampelo'), and a 'Gardar' button at the bottom.

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

A acción desenvolverase no momento que fagamos clic sobre o botón Gardar.

Centrémonos na parte do código referente á recollida da información das casillas de verificación. Unha vez validado o formulario créase un obxecto da clase Vacinacion chamado vacunacion. A clase Vacinacion unicamente define unha estrutura de datos para almacenar rexistros de valores nome – vacinacións aplicadas (consultar código da aplicación):

```
Vacinacion vacunacion=new Vacinacion(nome);
```

Empregando os métodos setter do obxecto vacunacion asignamos os valores das casillas de verificación aos atributos do obxecto. Hai que fixarse que o xeito de

recuperar o valor dunha casilla de verificación determinada é mediante o seu método `isSelected()`, o cal devolveranos `true` se a casilla esta marcada ou `false` se non o está.

```
//Recollida de datos dos JCheckBoxes
vacinacion.setTetano(chkTetano.isSelected());
vacinacion.setHepatite(chkHepatite.isSelected());
vacinacion.setSarampelo(chkSarampelo.isSelected());
```

Finalmente, este sería o código completo do `actionPerformed` do botón Gardar:

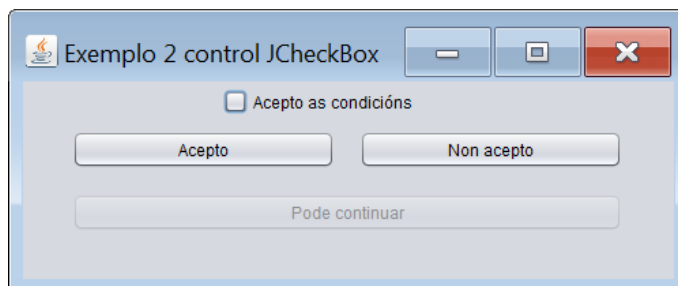
```
private void btnGardarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String nome=txtNome.getText().trim();
    if (nome.compareTo("")==0)
    {
        JOptionPane.showMessageDialog(this, "Debe indicar o nome");
        return;
    }
    //Nova Vacinacion
    Vacinacion vacinacion=new Vacinacion(nome);

    //Recollida de datos dos JCheckBoxes
    vacinacion.setTetano(chkTetano.isSelected());
    vacinacion.setHepatite(chkHepatite.isSelected());
    vacinacion.setSarampelo(chkSarampelo.isSelected());

    //Insertar o obxeto creado na BD
    //Non é parte desta unidade didáctica
    JOptionPane.showMessageDialog(this, "Registro gardado correctamente");
}
```

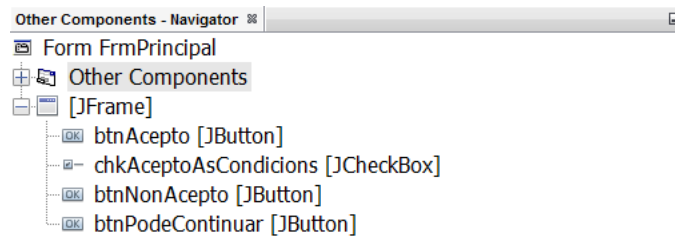
### Xestión de casillas de verificación empregando NetBeans (segundo exemplo)

No exemplo anterior vimos como recuperar os valores dunha casilla de verificación. Agora imos desenvolver un exemplo para explicar como asignar valores por código a unha casilla de verificación e como xestionar o seu evento `ItemStateChanged`. Para elo desenvolveremos o seguinte formulario:

A screenshot of a Java Swing window titled "Exemplo 2 control JCheckBox". The window has a standard title bar with minimize, maximize, and close buttons. Inside the window, there is a checkbox labeled "Acepto as condicións". Below the checkbox are two buttons: "Acepto" and "Non acepto". At the bottom of the window is a button labeled "Pode continuar", which appears to be disabled (grayed out).

No caso de que premamos sobre o botón Acepto marcarase a casilla de verificación. No caso de que premamos sobre o botón Non acepto desmarcarase a casilla de verificación. Ademais, cada vez que cambie o valor da casilla de verificación, xa sexa por acción directa do usuario sobre ela ou por acción indirecta a través do emprego dos botóns (Acepto e Non acepto) habilitarase ou deshabilitarase o botón Pode continuar en función do valor que teña a casilla de verificación.

O noso formulario quedará configurado do seguinte xeito:



Centrándonos no código da aplicación, imos implementar a primeira parte do programa, é dicir, a parte na que activamos ou desactivamos a casilla de verificación en función dos botóns premidos.

O código asociado ao botón acepto, o cal activa a casilla de verificación é o seguinte:

```
chkAceptoAsCondicions.setSelected(true);
```

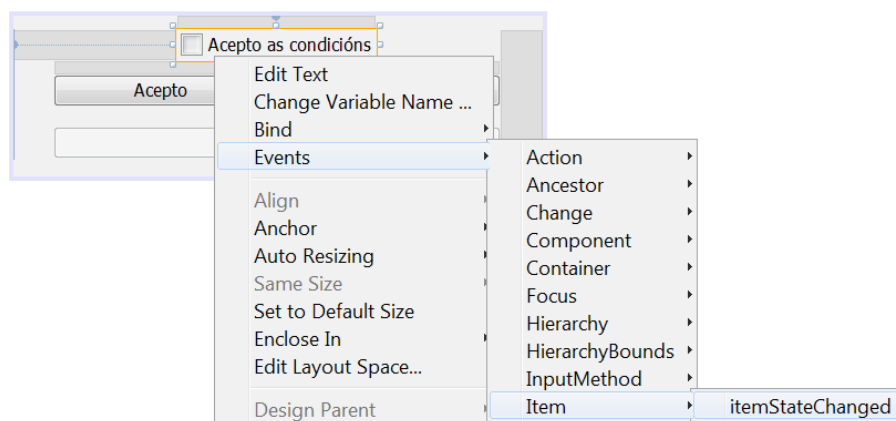
Como pódese observar, para activar unha casilla de verificación unicamente hai que pasarlle o valor true a través do seu método setSelected.

O código asociado ao botón Non acepto, o cal desactiva a casilla de verificación é o seguinte:

```
chkAceptoAsCondicions.setSelected(false);
```

Para desactivar unha casilla de verificación unicamente hai que pasarlle o valor false a través do seu método setSelected.

Respecto á segunda parte do programa, é dicir, a parte na que se habilita ou deshabilita o botón btnPodeContinuar en función do valor que toma a casilla de verificación cada vez que cambia (xa sexa este cambio debido á acción directa do usuario sobre a casilla de verificación ou pola súa acción indirecta a través do emprego dos botóns btnAcepto e btn NonAcepto), o xeito máis eficiente de resolvela é mediante o emprego dos eventos asociados á casilla de verificación, en concreto o evento ItemStateChanged, que é o evento disparado no momento que cambia o estado da casilla de verificación. Para facer isto debemos implementar o método que se disparará cando ocorra o evento ItemStateChanged da casilla de verificación chkAceptoAsCondicions. Para elo, pulsamos co botón dereito sobre a casilla de verificación chkAceptoAsCondicions e seleccionamos Events no menú despregable amosado. Dentro do submenú despregable que nos amosa Events seleccionamos Item e finalmente dentro do menú despregable que nos amosa Item seleccionamos a opción itemStateChanged:



Ao realizar esta acción o entorno de programación creará automaticamente o seguinte método:

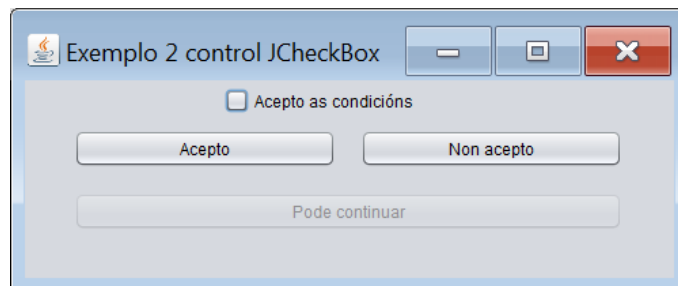
```
private void chkAceptoAsCondicionsItemStateChanged(java.awt.event.ItemEvent evt) {  
    // TODO add your handling code here:  
}
```

Agora o único que resta é implementar o seu código para que realice as accións que desexamos. Neste caso debemos escribir as seguintes liñas de código:

```
if (chkAceptoAsCondicions.isSelected())  
{  
    btnPodeContinuar.setEnabled(true);  
}  
else  
{  
    btnPodeContinuar.setEnabled(false);  
}
```

Cada vez que detectamos que cambia o estado da casilla de verificación (pola razón que sexa), comprobamos se no seu novo estado a casilla de verificación está marcada ou non. No caso de que estea marcada habilitamos o botón btnPodeContinuar e no caso contrario deshabilitamos o botón.

Se executamos a aplicación este será o resultado se a casilla de verificación non está marcada:



e este será o resultado no caso no que se detecte que a casilla de verificación está marcada:

