

Entorn gràfic

Introducció

Usarem el tutorial d'Oracle per a crear un GUI (*Graphical User Interface*) amb Swing, l'adreça del qual és

<https://docs.oracle.com/javase/tutorial/uiswing/index.html>

Quan treballem amb la consola, la interacció amb l'usuari és mitjançant text, en entorn visual, necessitem una o més finestres d'interacció on visualitzarem i llegirem informació, això és la interfície gràfica d'usuari (GUI).

A l'inici escriurem la lògica del projecte en el codi de la finestra de visualització, és l'estructura més simple, però és millor separar la lògica de la visualització en classes diferents.

Exemple resolt

Realitza l'apartat "Learning Swing with the NetBeans IDE" que està en l'adreça. Has de completar la lògica de la aplicació a soles.

<https://docs.oracle.com/javase/tutorial/uiswing/learn/index.html>

Contenidors d'alt nivell

Com has vist en el tutorial, has de crear un JFrame que és un contenidor d'alt nivell. Qualsevol component de la nostra interfície, per a poder apareixen en pantalla ha de pertànyer a l'arbre de components que comença en un contenidor d'alt nivell.

Tenim tres contenidors d'aqueix tipus:

- JFrame per a crear una finestra independent
- JDialog per a crear una finestra dependent d'una altra
- JApplet per a crear una finestra que s'incrustarà en una pàgina HTML

Nosaltres usarem un JFrame i JDialog, un o més, i no anem a usar JApplet.

En la pàgina següent pots veure com està construït un JFrame

<https://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html>

Un diàleg pot visualitzar-se de dues formes



- modal, l'execució del diàleg para l'execució en la finestra que l'ha obert i no es reprendrà mentre no es tanque el diàleg.
- no modal, l'execució del diàleg no para l'execució en la finestra que l'ha obert, per tant es pot continuar treballant tant en la finestra com en el diàleg.

En l'exemple següent es crea un diàleg modal (es posa `true` en el segon paràmetre del constructor), la crida al mètode `setVisible` és el que deté l'execució en la finestra que ha creat el diàleg. El mètode `setText` s'executa després de tancar el diàleg mitjançant un `dispose`.

```
NewJDialog diag = new NewJDialog(this, true);
diag.setVisible(true);
jLabel1.setText("Se ha cerrado");
```

Si no fora modal (es posa `false` en el segon paràmetre del constructor), llavors s'executaria el `setText` just després de visualitzar el diàleg.

```
NewJDialog diag = new NewJDialog(this, false);
```

Finestres internes

Existeix una classe `JInternalFrame` que permet crear finestres internes, però no és un contenidor d'alt nivell, però té pràcticament les mateixes funcionalitats que `JFrame`. Més informació en

<https://docs.oracle.com/javase/tutorial/uiswing/components/internalframe.html>

Diàlegs predefinits

Tenim diàlegs predefinits que ens ofereix la classe `JOptionPane` que ens permet traure missatges, o introduir informació des d'un diàleg. Els formats bàsics dels diàlegs són:

- `JOptionPane.showConfirmDialog`, per a un diàleg de confirmació
- `JOptionPane.showInputDialog`, per a un diàleg d'introducció
- `JOptionPane.showMessageDialog`, per a un diàleg de visualització d'un missatge
- `JOptionPane.showOptionDialog`, per a un diàleg d'elecció d'una opció

Tots tenen una versió `Internal` que s'usa quan el contenidor que l'obri és una finestra interna. Més informació en:

<https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html>

Més endavant, tens uns apartats on s'expliquen amb més profunditat.



Components

Tots els components de swing, el nom del qual comença amb J descendeixen de la classe JComponent, que al seu torn descendeix de Container. Tots els components tenen la capacitat de pintar-se a si mateix, de tractar esdeveniments, de contindre altres components.

Pots veure les propietats comuns de JComponent en

<https://docs.oracle.com/javase/tutorial/uiswing/components/jcomponent.html>

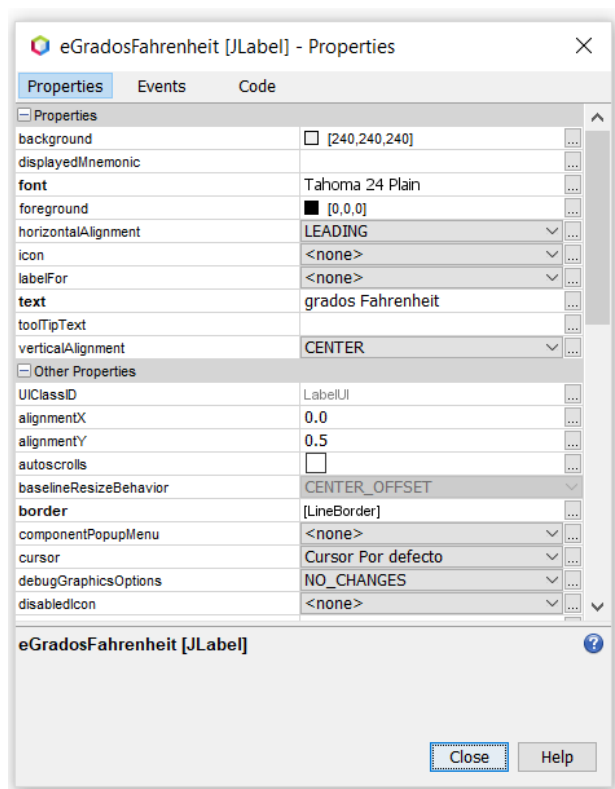
Cada component particular té les seues pròpies propietats. En la finestra **Properties** tenim les propietats del component seleccionat, en **Properties** tenim les propietats pròpies del component i en **Other Properties** tenim les propietats heretades de les classes anteriors.

En la finestra de propietats apareix el nom i el valor de la propietat.

Si el nom està en negreta, això vol dir que s'ha canviat el valor per defecte.

El valor es pot editar, canviant-lo directament en el requadre (valors simples, text, números) o clicant els ... que obri una finestra amb l'editor adequat per al valor de la propietat.

El valor de les propietats es pot obtenir des del codi amb el getter corresponent



```
String valor = jList.getSelectedValue(); // retorna el valor seleccionat d'una llista
String entrada = jTextField1.getText(); // retorna el text d'un camp de text
```

Es pot assignar el valor a una propietat amb el setter

```
jLabel1.setText("Hola món"); // assigna el text a una etiqueta
jLabel1.setIcon(new javax.swing.ImageIcon("simpsons.png")); // assigna una imatge a una etiqueta
```

A continuació es veuen diversos components i la seua utilització.

A part dels components oferits per Swing podem crear els nostres propis components, heretant de JComponent o d'un altre més especialitzat com JLabel o JPanel.

Podem afegir qualsevol component extern, simplement, afegint la llibreria que el maneja al projecte (per exemple SwingX).



GUI la interfície gràfica d'usuari

La interfície d'usuari permet obtenir informació de l'aplicació i oferir-la a l'usuari, i a l'usuari donar-li informació a l'aplicació, aquestes són operacions d'entrada o eixida.

La interfície d'usuari té dues parts, uns components que permeten l'entrada i eixida d'informació, i un sistema de control d'esdeveniments.

Els components permeten llegir i visualitzar informació.

Els esdeveniments permeten controlar les accions de l'usuari.

L'entorn en el qual es treballa (web, JavaFX, etc.) ens oferirà components diferents, nosaltres, treballarem amb els components que ens ofereix Swing.

La major part de les operacions amb la interfície suposen enviar text o rebre text, per tant és convenient manejar bé els mètodes de la classe `String`, i d'altres classes que transformen o formaten text.

Look and Feel

El Look and Feel o LAF defineix l'aspecte dels components que es van a pintar en el GUI.

El LAF s'aplica una única vegada a l'inici de l'aplicació.

Java proporciona els LAF següents, nom i classe

Metal	<code>javax.swing.plaf.metal.MetalLookAndFeel</code>
Nimbus	<code>javax.swing.plaf.nimbus.NimbusLookAndFeel</code>
CDE/Motif	<code>com.sun.java.swing.plaf.motif.MotifLookAndFeel</code>
Windows	<code>com.sun.java.swing.plaf.windows.WindowsLookAndFeel</code>
Windows Classic	<code>com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel</code>

En l'editor visual del GUI s'usa el LAF del sistema y en l'execució el Metal.

Per a assignar el LAF s'usa

```
UIManager.setLookAndFeel(className)
```

que necessita el nom de la classe associada al LAF

```
UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");
```

Cal tractar totes les excepcions que es llacen

```
try {
    UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
} catch (ClassNotFoundException | InstantiationException
        | IllegalAccessException | UnsupportedLookAndFeelException ex) {
}
```



La classe UIManager té uns mètodes que permeten recuperar els noms de les classes del LAF del sistema operatiu `getSystemLookAndFeelClassName` i el de Java (Metal) `CrossPlatformLookAndFeelClassName`,

```
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
```

El LAF es pot canviar durant l'execució de l'aplicació, però has de tenir en compte que això pot desbaratar l'aspecte del GUI.

En internet hi ha un munt de LAF, cal baixar la llibreria i afegir-la a l'aplicació.

Més informació en

<https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/index.html>

Eixida d'informació

L'eixida d'informació suposa passar informació **de memòria central a l'exterior**. En funció del receptor (una altra màquina, un humà) aqueixa informació es transforma de manera adequada. En el nostre cas la informació ha de transformar-se a un element visual comprensible per l'esser humà (text en el major dels casos).

JLabel, JTextField, JTextArea

Els components més habituals que usarem com a eixida d'informació seran

- JLabel que permet visualitzar una línia de text i/o una imatge
- JTextField que permet visualitzar una línia de text
- JTextArea que permet visualitzar diverses línies de text

Qualsevol component que tinga la propietat `text` visualitza el seu valor en pantalla.

```
jLabel1.setText("Hola món"); // assigna el text a una etiqueta  
jTextField1.setText("Hola món"); // assigna el text a un camp de text  
jTextArea1.setText("Hola món"); // assigna el text a una àrea de text
```

Com es pot apreciar el mètode `setText` és el mateix per als tres components (és lògic volem fer el mateix en els tres casos), el que canvia és la referència, és a dir, l'objecte on s'aplica l'acció. `setText` modifica el valor de text i per tant, modifica el text visualitzat pel component.

Cada component té capacitats diferents, per tant, cal triar el component més adequat per al que volem fer.

Si volem visualitzar més d'una línia de text hem d'utilitzar l'àrea de text

```
jTextArea1.setText("La suma és\n" + a + "+" + b + "=" + (a + b));
```

El `\n` provoca un canvi de línia. En aquesta sentència primer es concatenen tots els elements que hi ha en una cadena i aqueix valor s'assigna a l'àrea de text.



Una etiqueta està pensada per a visualitzar una única línia, però si usem HTML podem crear diverses línies amb
.

```
jLabel1.setText("<html>La suma és<br>" + a + "+" + b + "=" + (a + b) + "</html>");
```

tens més informació de l'ús d'HTML en

<https://docs.oracle.com/javase/tutorial/uiswing/components/html.html>

Si volem veure una imatge s'utilitza un JLabel i el seu mètode setIcon, cal crear un ImageIcon amb la imatge per a assignar-lo al JLabel

```
jLabel1.setIcon(new javax.swing.ImageIcon("simpsons.png"));
```

Si volem aplicar un format a la cadena que volem visualitzar podem usar el mètode format de la classe String. En els exemples següents, en el comentari, està el text resultant.

```
int a = 109;
jLabel1.setText(String.format("num = %d", a); // num = 109
jLabel1.setText(String.format("num = %+6d", a); // num = +109
jLabel1.setText(String.format("num = %X", a); // num = 6D
jLabel1.setText(String.format("car = %c", a); // car = m
double b = 32.147;
jLabel1.setText(String.format("num = %f", b); // num = 32,147000
jLabel1.setText(String.format("num = %E", b); // num = 3,214700E+01
jLabel1.setText(String.format("num = %5.2f", b); // num = 32,15
String t = "Hola";
jLabel1.setText(String.format("text = %s", t); // text = Hola
jLabel1.setText(String.format("text = %S", t); // text = HOLA
jLabel1.setText(String.format("text = %10s", t); // text = Hola
boolean x = true;
jLabel1.setText(String.format("bool = %b", x); // bool = true
```

Més informació en

<http://docs.oracle.com/javase/7/docs/api/java/util/formatter.html>

Java té moltes classes per a aplicar formats a dades d'eixida, número, dates, etc. Per als tipus d'informació més habituals tenim

- NumberFormat per a donar format a números
- DecimalFormat per a donar format a números decimals
- DateFormat per a donar format a dates i hora

Dins d'aqueixes classes hi ha altres més específiques.

Existeixen components més complexos que es poden usar per a traure informació per pantalla, per exemple: JTextPane, JEditorPane, JTable, JTree, etc.

<https://docs.oracle.com/javase/tutorial/uiswing/components/editorpane.html>

<https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>



<https://docs.oracle.com/javase/tutorial/uiswing/components/tree.html>

showMessageDialog

El `JOptionPane.showMessageDialog` obri un diàleg predefinit per a visualitzar un text. Té diferents formats

```
JOptionPane.showMessageDialog(parentComponent, message);  
JOptionPane.showMessageDialog(parentComponent, message, title, messageType);  
JOptionPane.showMessageDialog(parentComponent, message, title, messageType, icon);
```

`parentComponent` és la finestra pare del diàleg





`message` és el missatge a mostrar en el diàleg

`title` és el títol a mostrar en el diàleg

`messageType` és la icona predefinida del diàleg, la icona canvia en funció del LAF

`icon` és una pròpia per al diàleg, és una ruta a una imatge, si no es troba la imatge es mostra la icona predefinida

El `messageType` és un enter, però s'usen les constants definides en `JOptionPane`

`JOptionPane.ERROR_MESSAGE`, 
`JOptionPane.INFORMATION_MESSAGE`, 
`JOptionPane.PLAIN_MESSAGE`, no té icona
`JOptionPane.QUESTION_MESSAGE`, 
`JOptionPane.WARNING_MESSAGE`, 

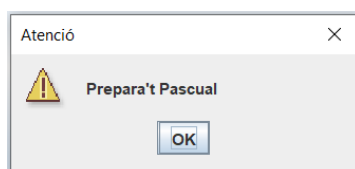
```
JOptionPane.showMessageDialog(ventana, "el número de vots ha de ser un enter");
```

`ventana` és el `JFrame` pare del diàleg i el missatge és *"el número de vots ha de ser un enter"*

```
JOptionPane.showMessageDialog(null, "el guanyador és\n" + nomJugador);
```

Aquest diàleg no té pare (val `null`) per tant es mostra centrat en la finestra, en el missatge tenim un `\n` que provoca un salt de línia en el text visualitzat en el diàleg.

```
JOptionPane.showMessageDialog(miVentana, "Prepara't Pascual", "Atenció", JOptionPane.WARNING_MESSAGE);
```



```
JOptionPane.showMessageDialog(null, "missatge", "títol", JOptionPane.QUESTION_MESSAGE,  
new javax.swing.ImageIcon(Provonova.class.getResource("es.png")));
```

Es crea una icona amb el fitxer *"es.png"* que està en el mateix paquet que la classe *Provonova*.



```
NomClasse.class.getResource(rutaImatge)
```

Aquesta és la forma de obtenir un recurs des d'un espai estàtic, com és el mètode main de la classe principal.

Entrada d'informació

L'entrada d'informació suposa passar informació **de l'exterior a memòria central**. Suposa una transformació de la informació, en el nostre cas, la informació la introduïrem en la finestra mitjançant el teclat o el ratolí, i després es transformarà i s'enviarà a memòria.

De manera general, qualsevol component que permet a l'usuari alterar el seu contingut es pot usar com a element d'entrada.

L'entrada d'informació depèn del component triat, encara que normalment haurem de transformar la informació que obtenim del component a la informació que usem en l'aplicació.

Text

TextField

El component que utilitzarem habitualment per a introduir informació és un `TextField`. Permet la introducció d'una sola línia de text.

TextArea

Si volem més d'una línia de text sense format podem usar un `TextArea`.

Tots dos components ens permet introduir text, cal posar el focus en el component i teclejar el text, després s'obté el text que conté el component amb el mètode `getText`.

```
String entrada = jTextField1.getText();
```

En l'exemple, s'obté el text que conté el camp de text anomenat `jTextField1` es crea un objecte de tipus cadena que s'assigna a la referència `entrada`, `getText` retorna un `String`, per tant, no necessita de cap mena de transformació.

Tenim dos components `JPasswordField` i `JFormattedTextField` que són una especialització del camp de text.

JPasswordField

`JPasswordField` permet introduir una clau (una paraula secreta) en la nostra aplicació. Aquest component no visualitza els caràcters que es teclegen. Té el mètode `getPassword` que retorna la informació introduïda en una matriu de caràcters, no useu `getText` que retorna un `String`, ja que es considera obsolet. L'ús d'una matriu de caràcters és més segur que l'ús d'un `String`, els valors de la matriu es podem matxucar



després d'usar-los, el valor d'un String es manté fins que el recupera el **garbage collector**.

<https://docs.oracle.com/javase/tutorial/uiswing/components/passwordfield.html>

tens un exemple i més informació.

JFormattedTextField

JFormattedTextField permet restringir l'entrada perquè se cenyisca a uns determinats criteris, que permeti llegir només enters, només double amb dos decimals, etc.

El mètode que permet recuperar el valor del camp és `getValue` que retorna un `Object`, per tant, cal fer-li un càsting al format desitjat. En la pàgina

<https://docs.oracle.com/javase/tutorial/uiswing/components/formattedtextfield.html>

tens més informació. En els apartats de llegir dades simples, veurem com s'usa un JFormattedTextField en alguns casos.

JList, JComboBox

Podem triar un text d'una llista o d'una llista desplegable, que usen JList i JComboBox respectivament. La diferència entre ells radica en el fet que la llista desplegable mostra només l'opció seleccionada i cal desplegar-la per a tindre accés a les altres opcions.

La llista d'opcions està en l'atribut `model`, cal canviar els valors per defecte Item 1, Item 2, Item 3, Item 4 pels valors que volem que apareguen en la llista, per defecte l'element seleccionat és el 0 (el primer)

jComboBox1 [JComboBox] - Properti... ×		
Properties	Events	Code
model		Item 1, Item 2, Item 3, Item 4
selectedIndex		0

El valor que s'obté és el de l'element seleccionat, `getSelectedValue` per a una llista i `getSelectedItem` per a un comboBox

```
jList1.getSelectedValue()  
jComboBox1.getSelectedItem()
```

hi ha altres mètodes que ens retornen informació de l'element seleccionat.

tens més informació en

<https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html>

<https://docs.oracle.com/javase/tutorial/uiswing/components/list.html>



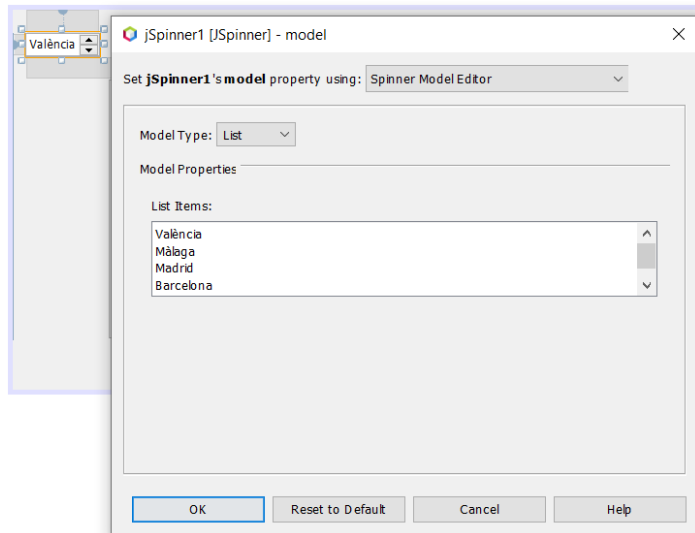
JSpinner

Podem usar un JSpinner per a triar un valor d'un conjunt usant fletxes. Té un model amb les dades que es poden seleccionar.

L'editor del model s'obri fent clic en els ... de la propietat

En l'exemple, el model és de tipus **List** i les valors que formen la llista estan en **List Items**, cada línia és una opció.

La manera d'obtindre el valor elegit en el spinner és mitjançant `getValue` que retorna un `Object`, per tant, cal fer un càsting del valor



```
String valor = (String) jSpinner1.getValue();
```

més en la pàgina

<https://docs.oracle.com/javase/tutorial/uiswing/components/spinner.html>

Número enter

JTextField

Amb un `JTextField` podem obtindre el text del camp, i després usar els mètodes `parseInt` o `valueOf` de la classe `Integer` per a transformen una cadena a un enter. La classe `Long` té els mètodes `parseLong` o `valueOf` per a transformar un text a long.

```
String entrada = jTextField1.getText();
int num = Integer.parseInt(entrada);
int val = Integer.valueOf("44");
long gran = Long.valueOf(entrada);
```

Si el text que es vol convertir no és un enter o un long, llavors no es realitza la transformació i es llança l'excepció `NumberFormatException` (això ocorre en els dos mètodes).

Als mètodes se'ls pot passar la base per a la transformació de la cadena, el número resultant està en base 10

```
num = Integer.valueOf("+52", 8); // num val 42
num = Integer.parseInt("473", 10); // num val 473
num = Integer.parseInt("Koala", 27); // num val 11109079
gran = Long.parseLong("23423654223452642465", 8); // gran val -351978705748968757
```



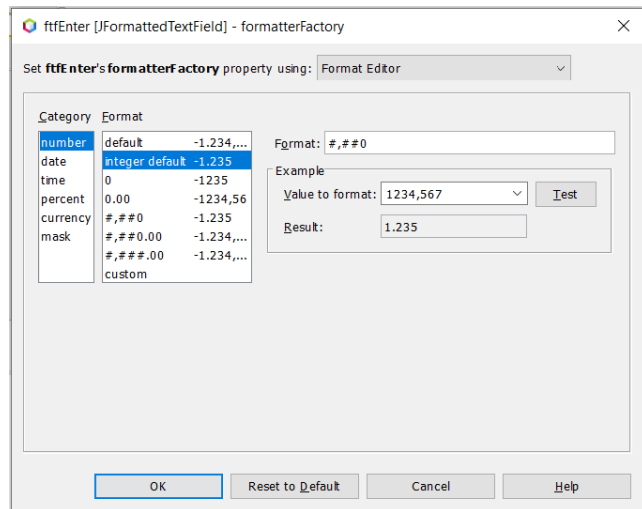
```
gran = Long.valueOf("de2fabaebafa9ca", 16); // gran val 1000637747954690506
gran = Long.valueOf("110011011010101011010001110", 2); // gran val 215658126
```

JFormattedTextField

Per al JFormattedTextField hem de seleccionar el tipus de format que volem aplicar. Per a això cal definir la propietat `formatterFactory`, en clicar els ... apareix la finestra de la dreta

Que ens permet triar la categoria de la informació i després el format, en l'exemple la categoria és `number` i el format és `integer default`.

A la dreta apareix el format de la màscara que s'aplica i un exemple de l'aplicació de la màscara.



La configuració anterior defineix el camp de text amb format següent

```
ftfEnter.setFormatterFactory(
    new javax.swing.text.DefaultFormatterFactory(
        new javax.swing.text.NumberFormatter(java.text.NumberFormat.getIntegerInstance()));
```

La referencia és `ftfEnter` i el mètode a cridar `getValue`, que retorna un Object de tipus `Number`, per tant, cal donar forma de `Number` al valor i després demanar l'enter amb `intValue`

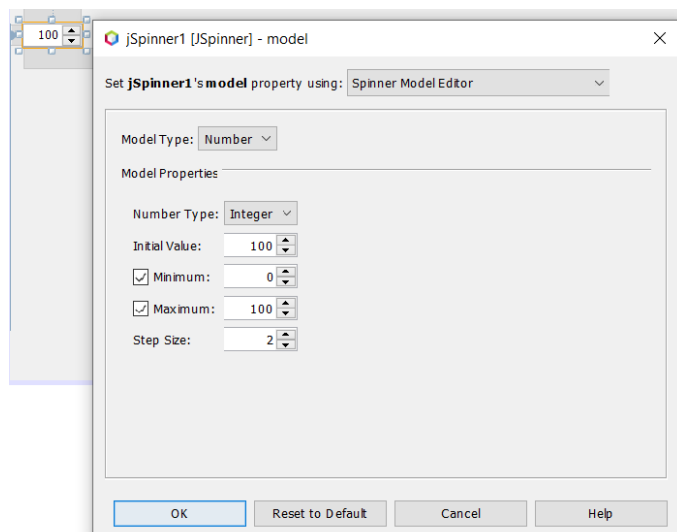
```
int importe = ((Number) ftfEnter.getValue()).intValue();
```

Els altres formats per a obtenir un enter són `0`, `#,##0` i `custom`. El format de `integer default` coincideix amb `#,##0`.

JSpinner

En el model del JSpinner es tria el tipus `Number` que manejarà números, poden ser de qualsevol tipus, per a manejar enters estan `Byte`, `Integer`, `Long` i `Short`.

En l'exemple de la dreta es defineix un conjunt de números enters, amb 100 com a valor inicial, 0 com a mínim, 100 com a màxim i es passa d'un valor al



següent o a l'anterior amb un pas de 2.

La manera d'obtenir el valor és mitjançant `getValue`, cal fer un càsting del valor

```
int valor = (int) jSpinner1.getValue();
```

JSlider

El component `JSlider` representa una barra amb una marca que llisca a sobre d'ella, el `JSlider` ens retorna el valor on es troba la marca sobre la barra.

Les propietats `maximum`, `minimum` i `value`, representen els valors màxim, mínim i inicial del `JSlider` el mètode `getValue` ens retorna el valor actual.

```
int num = jSlider1.getValue();
```

Si volem veure la escala cal posar un valor positiu a les propietats `majorTickSpacing`, `minorTickSpacing` i marcar la propietat `paintTicks`, si a més de l'escala volem els valors cal marcar la propietat `paintLabels`.

veure la pàgina

<https://docs.oracle.com/javase/tutorial/uiswing/components/slider.html>

Número amb decimals

JTextField

Amb un `JTextField` podem obtenir el text del camp i després usar la classe `Double` o `Float`, i els mètodes `parseDouble`, `parseFloat` o `valueOf`.

```
String entrada = jTextField1.getText();  
double num = Double.parseDouble(entrada);  
num = Double.valueOf("8844.4");  
float numf = Float.parseFloat(entrada);  
numf = Double.valueOf("-15.069");
```

Si el text que es vol convertir no és un `double` o `float`, llavors no es realitza la transformació i es llança l'excepció `NumberFormatException`.

Per a aquests mètodes, el punt és el separador dels decimals.

Només es poden transformar valors en base 10.

Podem usar notació científica per a representar els valors.

```
double val = Double.valueOf("1.234456e14");  
double var = Double.valueOf("-1.234456E-14");
```



JFormattedTextField

Usant JFormattedTextField podem seleccionar la categoria **number**, si es manté el **default** o s'elegeix **0.00**, **#,##0.00** o **custom**. es recupera un Number que cal adequar al tipus que volem

```
double num = ((Number) ftfNumero.getValue()).doubleValue();  
float numf = ((Number) ftfNumero.getValue()).floatValue();
```

en els dos casos es recupera el valor del camp de text i es passa a Number, després d'aqueix número s'obté el tipus que es vol.

Les categories **percent**, **currency** o **mask** retornen un número amb decimals, encara que cal tindre en compte que cadascun té les seues peculiaritats. Veure la pàgina

<https://docs.oracle.com/javase/tutorial/uiswing/components/formattedtextfield.html>

JSpinner

En el model del JSpinner es tria el tipus **Number** que manejarà números, poden ser de qualsevol tipus, per a manejar números amb decimals estan **Double** i **Float**.

Booleà

Els components JCheckBox i JRadioButton són els adequats per a obtenir un valor booleà mitjançant el mètode isSelected.

La diferència entre una casella de verificació (checkbox) i un botó de radio (radiobutton), és que els primers són independents (podem seleccionar els que vulguem) els segons són dependents (només podem seleccionar un).

Per als botons de radio necessitem afegir a la finestra un ButtonGroup, és un component que reuneix botons i només permetrà que un estiga seleccionat.

Cada radioButton pertanyent a un grup ha de posar el seu atribut buttonGroup amb el valor del grup, en codi es genera

```
buttonGroup1.add(jRadioButton1); // afig jRadioButton1 al grup buttonGroup1  
buttonGroup1.add(jRadioButton2); // afig jRadioButton2 al grup buttonGroup1
```

El component ButtonGroup no visualitza res en la finestra, i apareix en **Other Components** del navegador de la finestra.

En el disseny els components JCheckBox i JRadioButton tenen la propietat **selected** que indica si està seleccionat o no, en el codi el mètode setSelected amb els valors true o false realitza el mateix i en execució això es fa clicant el component.

```
jRadioButton2.setSelected(true); // jRadioButton2 està seleccionat  
jCheckBox1.setSelected(false); // jCheckBox1 no està seleccionat
```

Quan vulguem saber si està o no seleccionat s'usa el mètode isSelected



```

jCheckBox1.setSelected(true); // la casella està seleccionada
jCheckBox1.setText("Major d'edat"); // text associat a la casella
if (jCheckBox1.isSelected()) { // pregunta si la casella està seleccionada
    jTextArea1.append(" és major "); // afeg el text a l'àrea de text
}

```

El mètode `isSelected` retorna un valor booleà, no cal realitzar cap mena de transformació.

Els `JToggleButton` es poden utilitzar com a substituïts dels `JCheckBox` i `JRadioButton`, i ajuntar-los o no amb `ButtonGroup`. Els `JToggleButton` estan pensats per a representar disparadors de dues accions, per exemple validar o cancel·lar en funció de si el botó està seleccionat o no.

showConfirmDialog

El `JOptionPane.showConfirmDialog` obri un diàleg predefinit per a realitzar algun tipus de confirmació, el mètode retorna un enter que té un significat associat al botó usat per a tancar el diàleg.

Té els formats següents

```

JOptionPane.showConfirmDialog(parentComponent, message);
JOptionPane.showConfirmDialog(parentComponent, message, title, optionType);
JOptionPane.showConfirmDialog(parentComponent, message, title, optionType, messageType);
JOptionPane.showConfirmDialog(parentComponent, message, title, optionType, messageType, icon);

```

`parentComponent` és la finestra pare del diàleg

`message` és el missatge a mostrar en el diàleg

`title` és el títol a mostrar en el diàleg

`optionType` és els tipus d'opcions que ofereix el diàleg, els botons

`messageType` és la icona predefinida del diàleg, la icona canvia en funció del LAF

`icon` és una pròpia per al diàleg, és una ruta a una imatge, si no es troba la imatge es mostra la icona predefinida

L'únic nou és `optionType` que defineix els botons que mostra el diàleg, les constants estan definits en `JOptionPane`

`JOptionPane.OK_CANCEL_OPTION`, els botons [\[OK\]](#) i [\[Cancel\]](#)

`JOptionPane.DEFAULT_OPTION`, el botó [\[OK\]](#)

`JOptionPane.YES_NO_CANCEL_OPTION` els botons [\[Yes\]](#), [\[No\]](#) i [\[Cancel\]](#)

`JOptionPane.YES_NO_OPTION` els botons [\[Yes\]](#) i [\[No\]](#)

Les possibles respostes estan definides en les constants

`JOptionPane.CANCEL_OPTION`, s'ha prem el botó [\[Cancel\]](#)

`JOptionPane.CLOSED_OPTION`, s'ha prem el botó [\[x\]](#) que tanca el diàleg

`JOptionPane.NO_OPTION`, s'ha prem el botó [\[No\]](#)

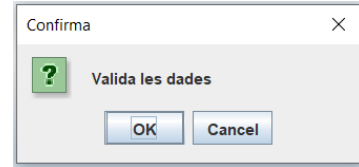
`JOptionPane.OK_OPTION`, s'ha prem el botó [\[OK\]](#)

`JOptionPane.YES_OPTION`, s'ha prem el botó [\[Yes\]](#)



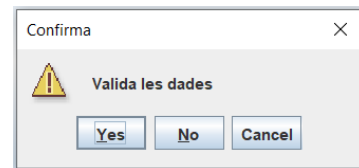
```
int entrada = JOptionPane.showConfirmDialog(null, "Valida les dades",
"Confirma",JOptionPane.OK_CANCEL_OPTION);
if (entrada == JOptionPane.OK_OPTION) {
    JOptionPane.showMessageDialog(null, "estas d'acord");
}
```

El diàleg no té cap JFrame pare, el missatge és “Valida les dades”, el títol del diàleg és “Confirma” i les opcions que es mostren són JOptionPane.OK_CANCEL_OPTION, el botó [OK] i [Cancel]. L’if comprova si s’ha clicat el botó [OK].



```
JOptionPane.showConfirmDialog(null, "Valida les dades", "Confirma", JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.WARNING_MESSAGE);
```

El diàleg és igual a l’anterior, però té la icona JOptionPane.WARNING_MESSAGE i el botons JOptionPane.YES_NO_CANCEL_OPTION, el diàleg és el de la dreta.



showInputDialog

El JOptionPane.showInputDialog obri un diàleg predefinit que permet l’entrada d’un text, retorna una cadena de text o null si es cancel·la l’entrada (clicar el botó [Cancel] o tancar el diàleg).

Té els formats següents

```
JOptionPane.showInputDialog(message);
JOptionPane.showInputDialog(parentComponent, message);
JOptionPane.showInputDialog(message, initialValue);
JOptionPane.showInputDialog(parentComponent, message, initialValue);
JOptionPane.showInputDialog(parentComponent, message, title, messageType);
JOptionPane.showInputDialog(parentComponent, message, title, messageType, icon, selectionValues,
initialSelectionValue);
```

El selectionValues és una matriu de valors possibles (Object) que es mostren en el diàleg

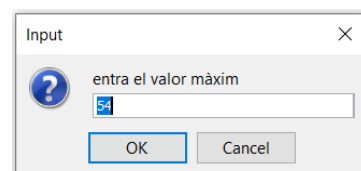
El initialSelectionValue és el valor inicial (Object) que es mostra en el diàleg

```
String entrada = JOptionPane.showInputDialog(finestra, "entra el valor màxim");
if (entrada != null) {
    JOptionPane.showMessageDialog(finestra, "el número és " + entrada);
}
```

finestra és el JFrame pare del diàleg i el missatge és “entra el valor màxim”. L’if comprova si no s’ha cancel·lat l’entrada, és a dir, que entrada no és null.

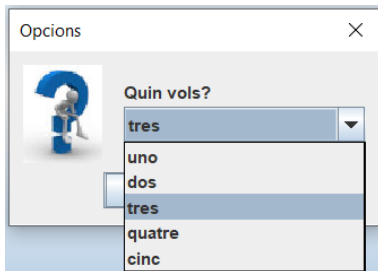
```
String entrada = JOptionPane.showInputDialog(finestra, "entra el valor màxim", 54);
```

El diàleg mostra el valor inicial és “54” en el camp d’entrada.



Hi ha una versió en la qual poden definir els conjunt de possibles valors a introduir, retorna un Object o null.

```
String[] ops = {"uno", "dos", "tres", "quatre", "cinc"};
ImageIcon icon = new javax.swing.ImageIcon(getClass().getResource("/img/pregu.jpg"));
Object opcio = JOptionPane.showInputDialog(finestra, "Quin vols?", "Opcions",
    JOptionPane.INFORMATION_MESSAGE, icon, ops, "tres");
if (opcio != null) {
    JOptionPane.showMessageDialog(finestra, "has seleccionat " + opcio);
}
```



ops és la matriu d'opcions, "tres" és el valor per defecte, icon és la icona que es visualitzarà, si no existeix es mostra la icona per defecte,

Retorna el valor seleccionat de la llista desplegable.

showOptionDialog

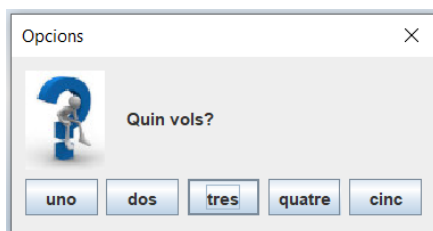
showOptionDialog retorna un enter que representa l'índex de la matriu d'opcions o el fet d'haver tancar el diàleg, el seu format és

```
JOptionPane.showOptionDialog(parentComponent, message, title, optionType, messageType, icon, options,
    initialValue);
```

El options és una matriu de valors possibles (Object) que es mostren en el diàleg

El initialValue és el valor inicial (Object) que es mostra en el diàleg

```
String[] ops = {"uno", "dos", "tres", "quatre", "cinc"};
ImageIcon icon = new javax.swing.ImageIcon(getClass().getResource("/img/pregu.jpg"));
int opcio = JOptionPane.showOptionDialog(finestra, "Quin vols?", "Opcions",
    JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, icon,
    ops, "tres");
if (opcio != JOptionPane.CLOSED_OPTION) {
    JOptionPane.showMessageDialog(finestra, "has seleccionat " + ops[opcio]);
}
```



ops és la matriu d'opcions, es mostra un botó per a cada opció.

icon és la icona que es visualitzarà, si no existeix es mostra la icona per defecte

finestra és el JFrame pare del diàleg i el missatge és "Quin vols?", el títol del diàleg és "Opcions", el tipus de opcions, el tipus de missatge (això defineix la icona per defecte), la icona, la matriu d'opcions i "tres" és l'opció per defecte.



Quan es clica una opció es tanca el diàleg. L'if comprova si no s'ha clicat el botó de tancament del diàleg .

Esdeveniments

Per a poder realitzar les nostres aplicacions necessitem controlar les coses que passen en la finestra, és a dir, els esdeveniments que es produeixen en ella.

Cada component genera un conjunt d'esdeveniments. Hi ha molts tipus d'esdeveniments; d'acció, de focus, de teclat, de ratolí, de canvi, etc.

Cada tipus d'esdeveniment té un oïdor (*listener* en anglés): ActionListener, MouseListener, KeyListener, FocusListener, CaretListener, ListSelectionListener, etc. També, es pot usar un Adapter que proporciona una implementació per defecte dels mètodes del oïdor i sols es sobreescriu que ens interessa, FocusAdapter, KeyAdapter, MouseAdapter, WindowAdapter, etc.

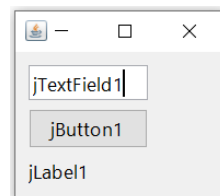
Hem d'afegir al component un oïdor per cada tipus d'esdeveniment que volem controlar, en l'oïdor s'implementa la resposta a l'esdeveniment.

En el menú contextual d'un component en **Events** apareix la llista de tots els esdeveniments que aquest component pot generar.

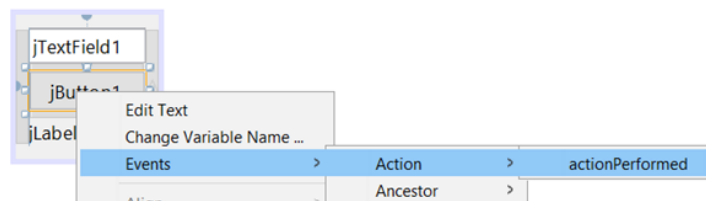
Els components generen tots els esdeveniments que tenen associats, només cal tractar els esdeveniments útils per a la resolució del nostre problema

Treballarem, majoritàriament, amb l'esdeveniment d'acció (ActionEvent). És l'esdeveniment que es genera quan es realitza l'acció lògica d'aqueix component, per exemple, fer clic sobre un botó o una casella de selecció, o prémer retorn de carro en un camp de text. No tots els components tenen l'esdeveniment d'acció.

Volem fer una l'aplicació que visualitza el valor del camp de text en l'etiqueta en majúscules quan es prem el botó. En la finestra hi ha un botó (la seua referència és jButton1), una etiqueta (jLabel1) i un camp de text (jTextField1).



Oïrem i respondrem a l'esdeveniment d'acció que és l'acció lògica de prémer el botó.



L'esdeveniment es llança en fer clic en el botó o prémer la barra espaiadora quan el botó està seleccionat.

Es llacen més esdeveniments (de ratolí, de focus, etc.), però els ignorarem.



Creem un oïdor d'acció i l'afegim al botó, per a això, cliquem el botó dret sobre el component i triem **Events > Action > actionPerformed**, que crea el codi següent

1. l'assignació de l'oïdor al component `jButton1.addActionListener`
2. la creació d'un objecte oïdor d'acció `new java.awt.event.ActionListener`

```
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
```

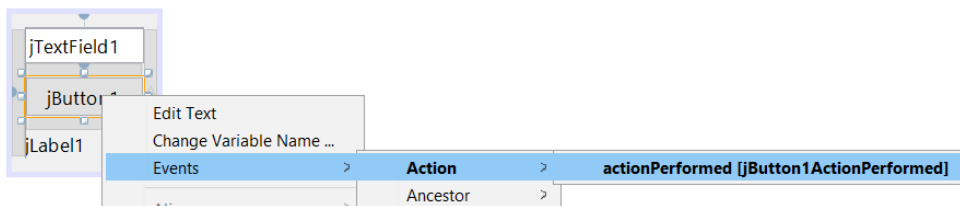
L'objecte oïdor només té un mètode (`actionPerformed`) que és on s'escriu el codi gestor de l'esdeveniment. En l'exemple, el mètode `actionPerformed` crida al mètode `jButton1ActionPerformed`, que és on escrivim el codi de resposta a l'esdeveniment.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String entrada = jTextField1.getText();
    jLabel1.setText(entrada);
}
```

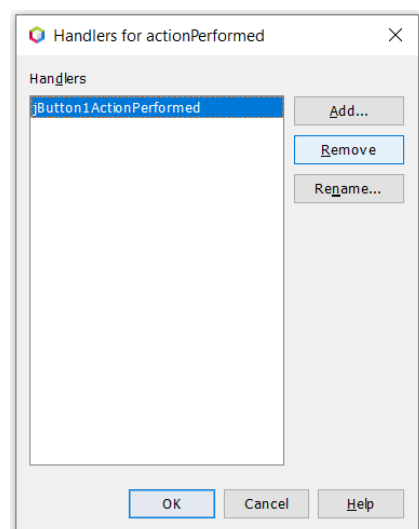
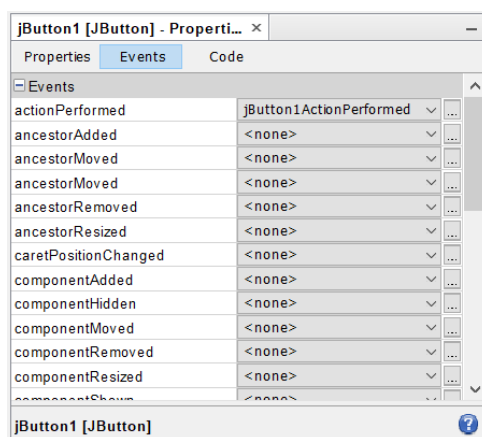
El codi que hem escrit obté el text que hi ha en el camp de text `jTextField1` i ho assigna a l'etiqueta `jLabel1`. Excepte aqueixes dues línies de codi, la resta s'ha generat de manera automàtica.

El mètode té el paràmetre, `evt`, és un objecte que conté informació de l'esdeveniment d'acció que s'ha produït, i que podem utilitzar.

Quan un component té un oïdor creat a l'entorn de disseny, aquest apareix en negreta en la llista d'esdeveniments.



En la finestra de **Properties** en la pestanya **Events** tens tots els esdeveniments del component seleccionat i si té o no algun oïdor associat.



En fer clic en els punts suspensius ... apareix una finestra on pots afegir, llevar o canviar de nom algun gestor de l'esdeveniment ([Handler](#)).

L'etiqueta no té l'esdeveniment d'acció, anem a usar l'esdeveniment de ratolí fer clic.

Seleccionem l'etiqueta i triem [Events > Mouse > mouseClicked](#), que crea el codi

```
jLabel1.addMouseListener(new java.awt.event.MouseAdapter() {  
    public void mouseClicked(java.awt.event.MouseEvent evt) {  
        jLabel1MouseClicked(evt);  
    }  
});  
private void jLabel1MouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
}
```

A l'etiqueta s'afegeix un oïdor de ratolí (addMouseListener) i el gestor és un MouseAdapter, es defineix el mètode mouseClicked que s'executa quan es clica l'etiqueta.

El MouseListener té cinc mètode mouseClicked, mouseEntered, mouseExited, mousePressed i mouseReleased, si es crea un MouseListener cal implementar els cinc mètodes, però un MouseAdapter ja té els mètodes implementats per defecte, per tant, només cal sobreescrivre el que ens interessa.

Cal escriure el codi de resposta a l'esdeveniment de fer clic en l'etiqueta

```
private void jLabel1MouseClicked(java.awt.event.MouseEvent evt) {  
    jLabel1.setText(jLabel1.getText().toUpperCase());  
}
```

Més d'un gestor o oïdor

La resposta a un esdeveniment pot tindre més d'un gestor ([Handler](#)), aquests s'executen en l'ordre d'escriptura.

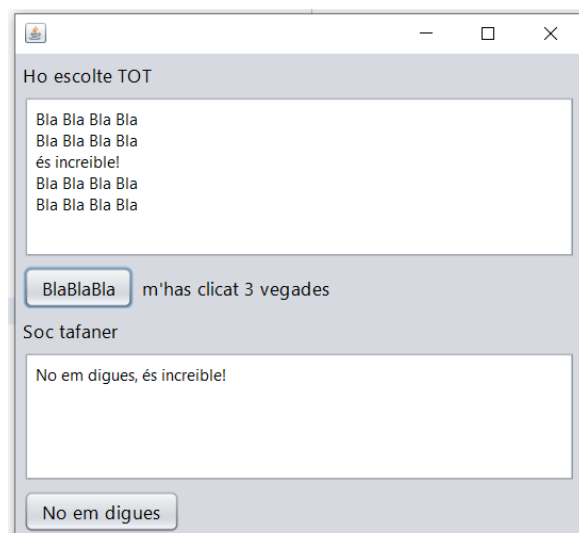
Un esdeveniment pot ser oït per més d'un oïdor i un oïdor pot oir esdeveniments de components diferents.

En l'exemple, tindrem un botó amb dos gestors i altre amb dos oïdors.

En la finestra tenim dues àrees de text, dos botons i tres etiquetes.

Quan es clica el botó [\[BlaBlaBla\]](#) s'escriu "Bla Bla Bla Bla" en l'àrea de text superior i es visualitza el nombre de vegades que s'ha clicat el botó.

Quan es prem el botó [\[No em digues\]](#) s'escriu "No em digues, és increïble!" en



la àrees de baix i en la de dalt “*Bla Bla Bla Bla és increïble!*”.

El botó [BlaBlaBla] té un oïdor amb dos gestors escriuDalt i incrementa, el primer escriu en l'àrea de text superior i el segon canvia el comptador i el mostra quan es prem el botó.

El botó [No em digues] té dos oïdors amb el seu gestor corresponent.

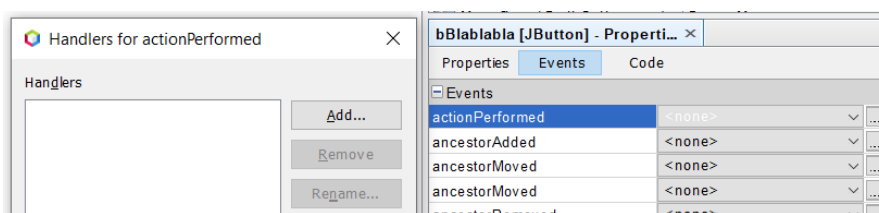
Els components que es manejaran en el codi es canvien de nom:

- l'àrea de text superior es diu taDalt
- l'àrea de text inferior es diu taBaix
- el botó de dalt es diu bBlablaba
- l'etiqueta que visualitza el comptador es diu eComptador
- el botó de baix es diu bNoemdigues

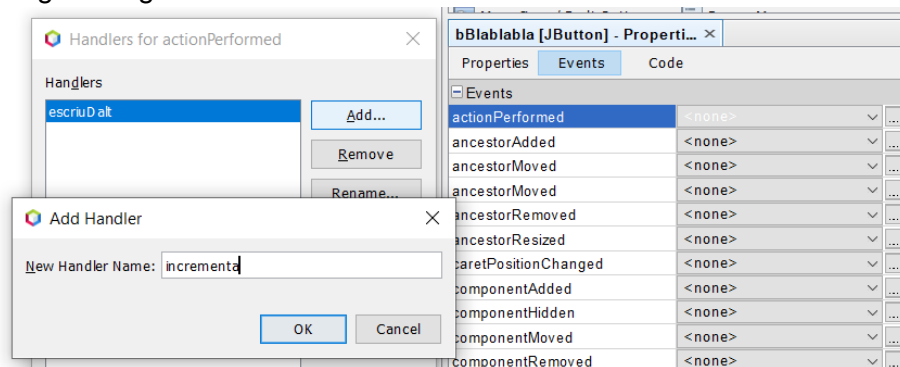
Les primeres lletres en minúscules representen el tipus de component ta de TextArea, b de botó i e de etiqueta, després hi ha un nom associat al significat del component en el codi.

Creem els gestors, hi ha tres escriuDalt, escriuBaix i incrementa, són tres mètodes que ha de tenir l'esdeveniment com a paràmetre.

Es selecciona el botó bBlablaba i s'obri la seua finestra d'esdeveniments



Afegim els gestors



Es creen els mètodes, cal afegir el codi.

```
private void escriuDalt(java.awt.event.ActionEvent evt) {  
    taDalt.append("Bla Bla Bla Bla\n");  
    taDalt.setCaretPosition(taDalt.getDocument().getLength());  
}  
  
private void incrementa(java.awt.event.ActionEvent evt) {  
    String[] paraules = eComptador.getText().split(" ");
```



```
int comptador = Integer.parseInt(paraules [paraules.length - 2]);
comptador++;
eComptador.setText("m'has polsat " + comptador + " vegades");
}
```

En afegir els gestors es crea o modifica l'oïdor d'acció del component

```
bBlabla.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        escriuDalt(evt);
        incrementa(evt);
    }
});
```

El mètode actionPerformed de l'oïdor crida els dos gestors que hem afegit.

Per al botó bNoemdigues hem de crear els oïdors d'acció, les referències són ac1 i ac2

```
java.awt.event.ActionListener ac1 = new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        String textBotó = ((JButton) (evt.getSource())).getText();
        taBaix.append(textBotó + ", és increïble!\n");
        taBaix.setCaretPosition(taBaix.getDocument().getLength());
    }
};
java.awt.event.ActionListener ac2 = new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        taDalt.append("és increïble!\n");
        escriuDalt(evt);
    }
};
```

després s'han d'assignar al botó

```
bNoemdigues.addActionListener(ac1);
bNoemdigues.addActionListener(ac2);
```

Si un oïdor d'acció s'afegeix dues vegades, llavors s'executa dues vegades.

Es pot eliminar un oïdor d'acció

```
bNoemdigues.removeActionListener(ac2);
```

En la pàgina següent, teniu com escriure oïdors per a tota mena d'esdeveniments

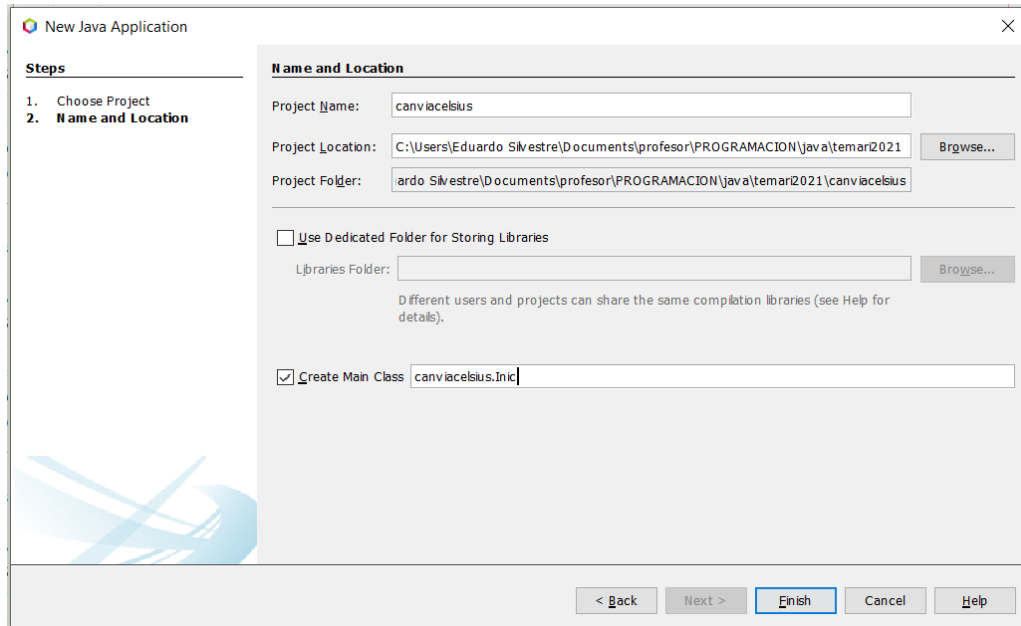
<https://docs.oracle.com/javase/tutorial/uiswing/events/index.html>

Exercici 1

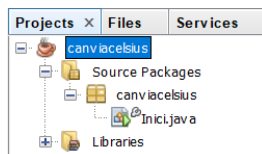
Crea una aplicació que converteix graus Celsius a graus Fahrenheit, usant la fórmula $F = (9 * C / 5) + 32$ on F són els graus Fahrenheit i C els graus Celsius



Crea una aplicació nova amb classe principal i main. El projecte es diu canviacelsius i la classe principal Inici.



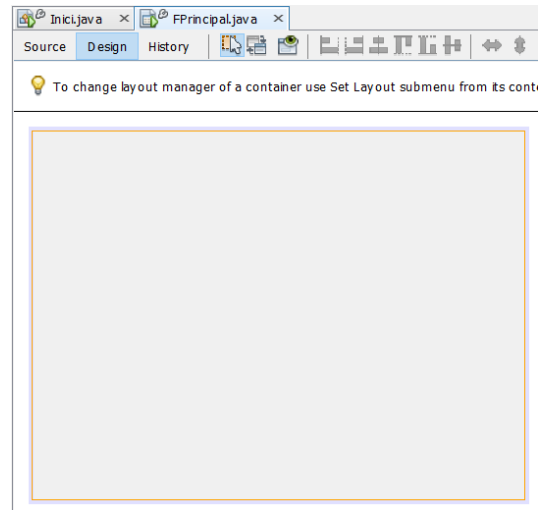
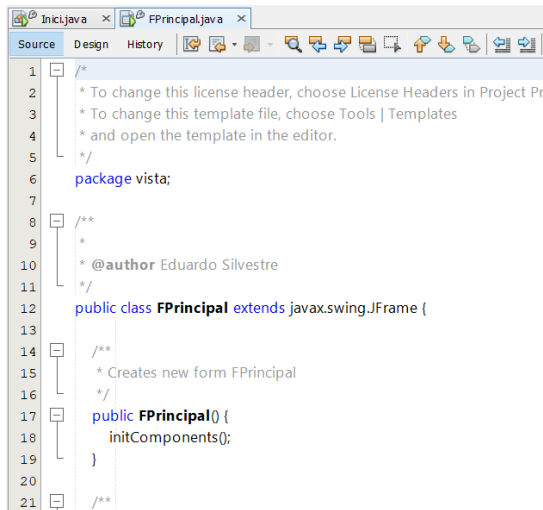
L'estructura del projecte creat és el següent



Afegeix el paquet vista que contindrà la nostra finestra de visualització. Per a això, sobre el paquet vista amb el botó dret selecciona **New > JFrame Form...** per a crear la classe FPrincipal que permetrà crear la Interfície Gràfica d'Usuari de l'aplicació.

L'entorn ens ofereix un editor de text per al codi de la classe i un editor gràfic per a construir el GUI.





Totes les accions que es fan en l'editor gràfic es transformen en codi, a més aqueix codi no es pot modificar en l'editor de codi. **El codi sobre fons gris no es pot modificar,** es pot veure i copiar.

La classe principal Inici només servirà per a iniciar l'aplicació, és a dir, crear l'objecte finestra (new) i mostrar-lo (setVisible).

```
package canviacelsius;
import vista.FPrincipal;
public class Inici {
    public static void main(String[] args) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FPrincipal().setVisible(true);
            }
        });
    }
}
```

Les aplicacions Swing han de controlar els esdeveniments, per tant, es creen sobre el fil de tractament de la cua d'esdeveniments (EventQueue) es crea un objecte Runnable sobre aquest fil que s'encarrega de crear la finestra principal.

Altra forma de crear la finestra principal és

```
javax.swing.SwingUtilities.invokeLater(new Runnable() {
    public void run() { new FPrincipal().setVisible(true); }
});
```

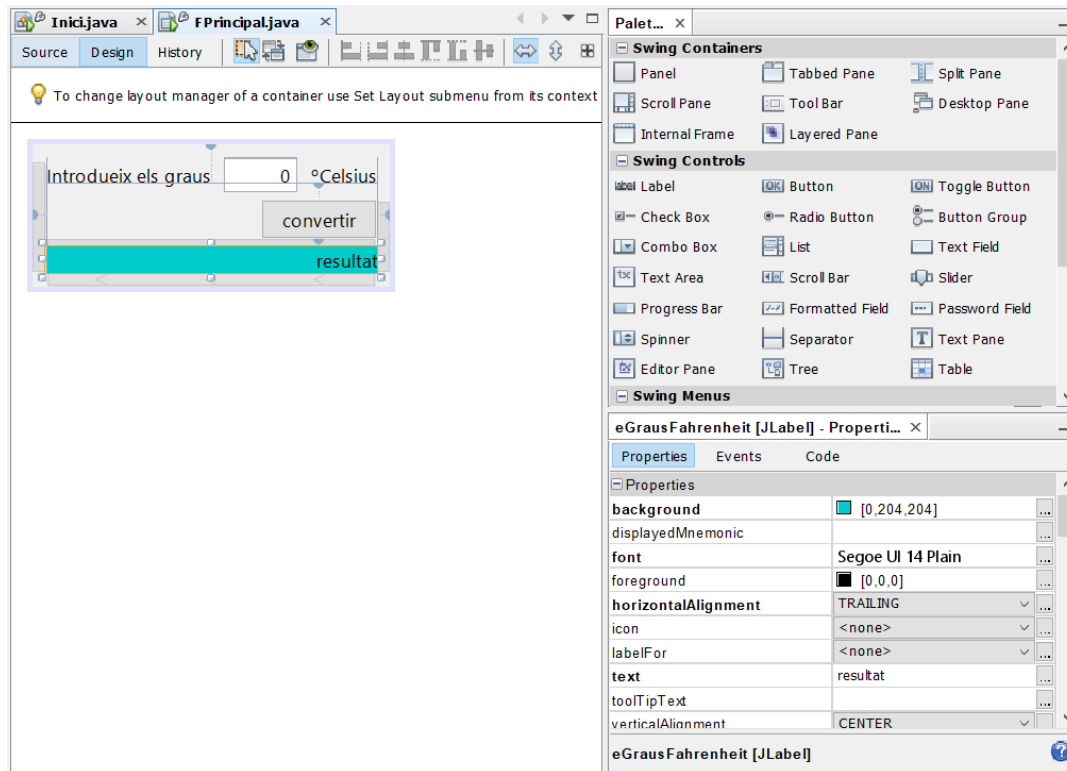
En la finestra hi ha dues etiquetes de text fix "Introdueix els graus" i "°Celsius", un camp de text per a introduir els graus Celsius, un botó per a executar la conversió i una etiqueta per a veure el resultat de la conversió a graus Fahrenheit.

Canvia els noms de les referències dels components que s'usen en el codi: bConvertir per al botó, tfGausCelsius per al camp de text i eGausFahrenheit per a l'etiqueta.



Els components s'agafen de la finestra **Palette** (la paleta de components), fes clic sobre un component per a seleccionar-lo i després fes clic en la zona de disseny per a crear-lo i afegir-lo a la finestra.

En la zona de disseny pots actuar sobre el component per a moure'l o per a canviar la seua grandària.



Quan selecciones un component, en la finestra **Properties**, es visualitzen les seues propietats, es poden canviar per a adequar el component a les nostres necessitats. Component i propietats modificades

jLabel1: **font** Segoe UI 14 Plain, **text** Introdueix els graus

jLabel2: **font** Segoe UI 14 Plain, **text** °Celsius

tfGausCelsius: **font** Segoe UI 14 Plain, **text** 0, **horizontalAlignment** TRAILING

bConvertir: **font** Segoe UI 14 Plain, **text** convertir

eGausFahrenheit: **font** Segoe UI 14 Plain, **text** resultat, **background** (0, 204, 204), , **opaque** true, **horizontalAlignment** TRAILING

Prova amb altres valors, per a vore els canvis.

Tots el que fas en la zona de disseny i en les propietats es veu reflectit en el codi, és el codi sobre fons gris. El codi sobre fons gris no es pot modificar en la finestra de l'editor. Hi ha dues zones: la definició de les referències dels components i el mètode



initComponents amb la creació i configuració inicial dels objectes. Davant de cada zona de codi hi ha un comentari indicant que aquest codi no es pot modificar.

```
// Variables declaration - do not modify
private javax.swing.JButton bConvertir;
private javax.swing.JLabel eGrausFahrenheit;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JTextField tfGrausCelsius;
// End of variables declaration
```

El mètode initComponents

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    eGrausFahrenheit = new javax.swing.JLabel();
    bConvertir = new javax.swing.JButton();
    tfGrausCelsius = new javax.swing.JTextField();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    jLabel1.setText("Introdueix els graus");

    jLabel2.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    jLabel2.setText("°Celsius");

    eGrausFahrenheit.setBackground(new java.awt.Color(0, 204, 204));
    eGrausFahrenheit.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    eGrausFahrenheit.setHorizontalAlignment(javax.swing.SwingConstants.TRAILING);
    eGrausFahrenheit.setText("resultat");
    eGrausFahrenheit.setOpaque(true);

    bConvertir.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    bConvertir.setText("convertir");

    tfGrausCelsius.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    tfGrausCelsius.setHorizontalAlignment(javax.swing.JTextField.TRAILING);
    tfGrausCelsius.setText("");

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
                .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
                .addComponent(eGrausFahrenheit, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
                .addComponent(bConvertir, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
                .addComponent(tfGrausCelsius, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
            )
            .addContainerGap(150, true)
        )
    );
}
```



```

        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(tfGrausCelsius, javax.swing.GroupLayout.PREFERRED_SIZE,
                    55, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel2)
                .addGap(0, 0, Short.MAX_VALUE))
            .addComponent(eGrausFahrenheit, javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(bConvertir)))
        .addContainerGap()
    };
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel1)
                .addComponent(jLabel2)
                .addComponent(tfGrausCelsius, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(bConvertir)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(eGrausFahrenheit)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

    pack();
} // </editor-fold>

```

El mètode `initComponents` es crida des del constructor de la finestra

```

public FPrincipal() {
    initComponents();
}

```

L'únic esdeveniment que controla l'aplicació és prémer el botó, és un esdeveniment d'acció.

Crea l'esdeveniment d'acció, selecciona el botó, fes clic amb el botó dret i tria [Events > Action > actionPerformed](#) que crea el codi següent en el mètode `initComponents`

```

bConvertir.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        bConvertirActionPerformed(evt);
    }
});

```



i el mètode que respon a l'esdeveniment

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

La línia amb el comentari // TODO add your handling code here: es substitueix pel codi que respon a l'esdeveniment. **No esborres el comentari, com a primer pas, si es junten les línies grises no podràs escriure el codi.**

La lògica de la resposta a prémer el botó és la següent

1. Obtindre el contingut del camp de text tfGrausCelsius, un text amb els graus Celsius
2. Transformar el text a un double
3. Realitzar la transformació de graus Celsius a graus Fahrenheit
4. Visualitzar la transformació en l'etiqueta eGrausFahrenheit

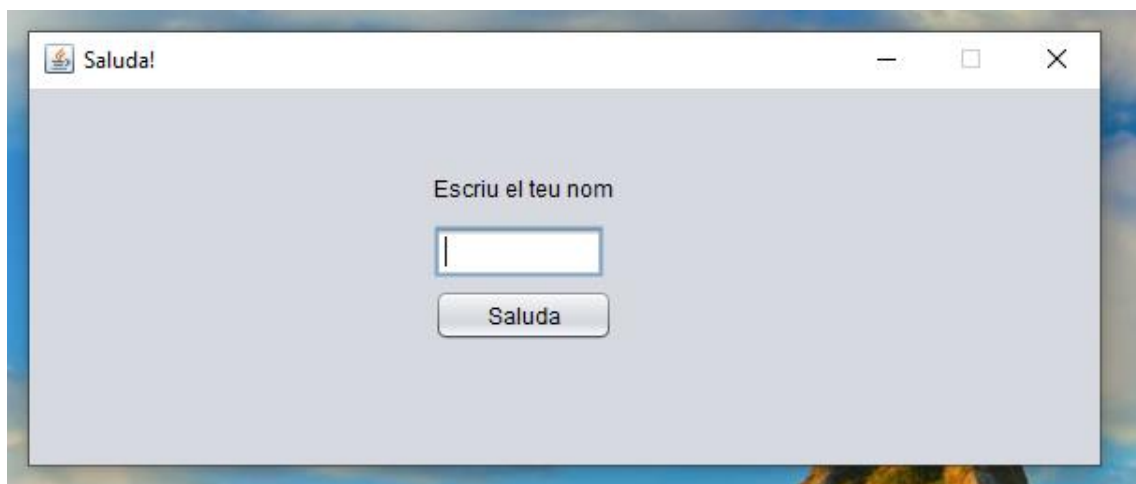
El mètode queda de la forma següent

```
private void bConvertirActionPerformed(java.awt.event.ActionEvent evt) {  
    String gradosC = tfGrausCelsius.getText();  
    double grausCelsius = Integer.parseInt(gradosC);  
    double grausFahrenheit = (9 * grausCelsius / 5) + 32;  
    eGrausFahrenheit.setText(grausCelsius + "°C = " + grausFahrenheit + "°F");  
}
```

El projecte ja està complet, pots executar-lo.

Exercici 2

Has de crear una finestra en la que es demane un nom i al fer click en el butó "Saludar", aparega un diàleg, del tipus JOptionPane, en què es mostre la salutació al nom introduït. Una proposta de disseny és la següent:

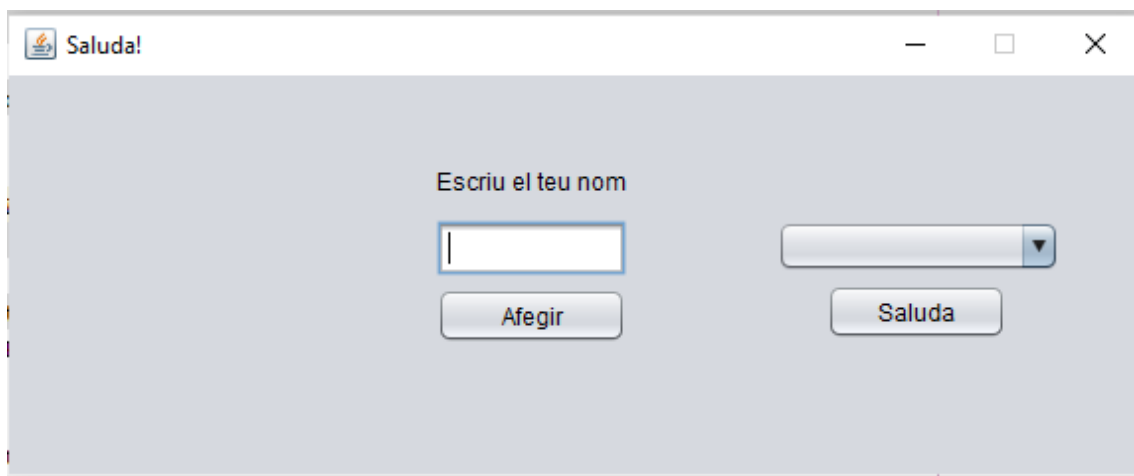




Has de posar un títol a la ventana i establir que no pot ser redimensionable. A més, en la propietat `preferredSize`, sel·leccionar l'opció per a què la grandària siga la mateixa que es veu al editor.

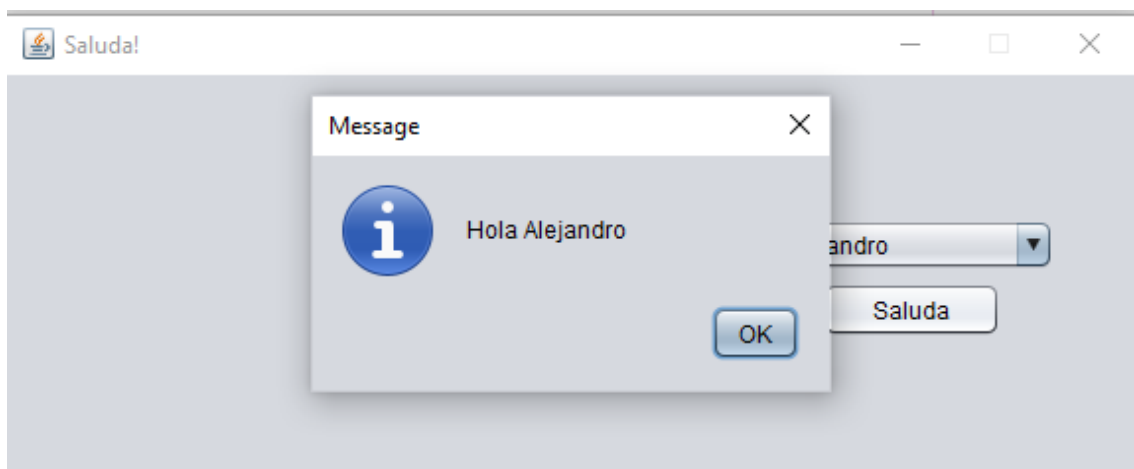
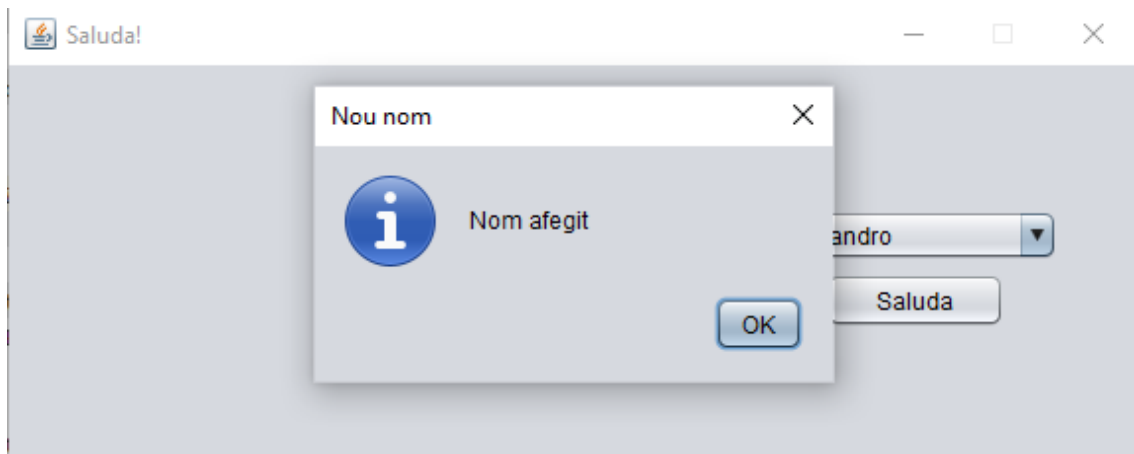
Exercici 3

Tomant com a base l'exercici anterior, l'ampliarem amb la introducció d'un nou element 'combobox'. Ara tindrem un butó per afegir el nom al Combo Box i mantindrem el de salutar:

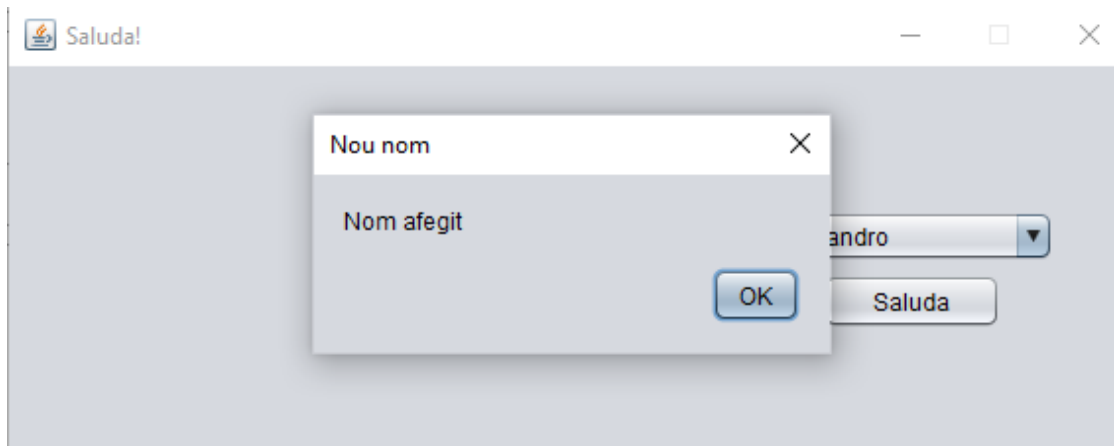


S'aniran afegint elements a la llista i després podrem triar a quin salutar.

Has de crear dos diàlegs, un per a mostrar que s'ha afegit el nom i altre mantenint la salutació.



Ara configura l'ícona de l'última imatge per a què aparega sense:



Les finestres deben obrirse al mig de la pantalla, configura-ho.

Exercici 4

En el següent exercici es va a un formulari amb els següents components:

- JTextField



- JRadioButton
- JCheckBox
- JTextArea
- JSpinner

A més, tras enviar el formulari, apareixerà un diàleg amb que podrem confirmar les dades introduïdes. Fixat que també hem introduït un component nou, anomenat separador.

HAS DE CANVIAR LA FONT I EL TAMANY OBLIGATORIAMENT, A MÉS ESTABLIR ALGUN CAMP AMB COLOR DE FONTS.

En primer lloc, disposa els components tal i com veus a la imatge següent:

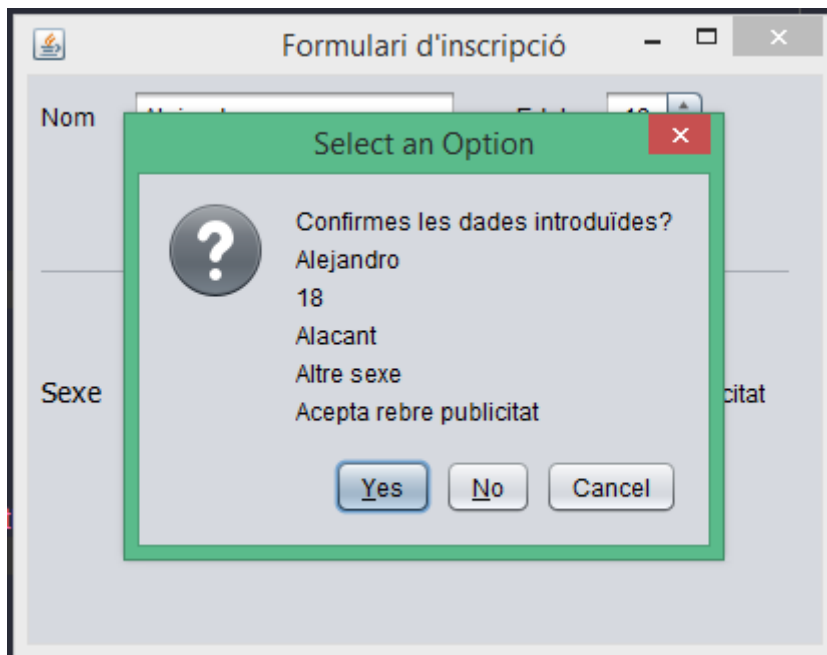
Canvia els noms dels components per tal de que siga més fàcil recordarlos.

Després, hem de conseguir que no es seleccionen totes les opcions dels radio button. Cerca en la xarxa sobre el funcionament d'aquest element i soluciona el problema.

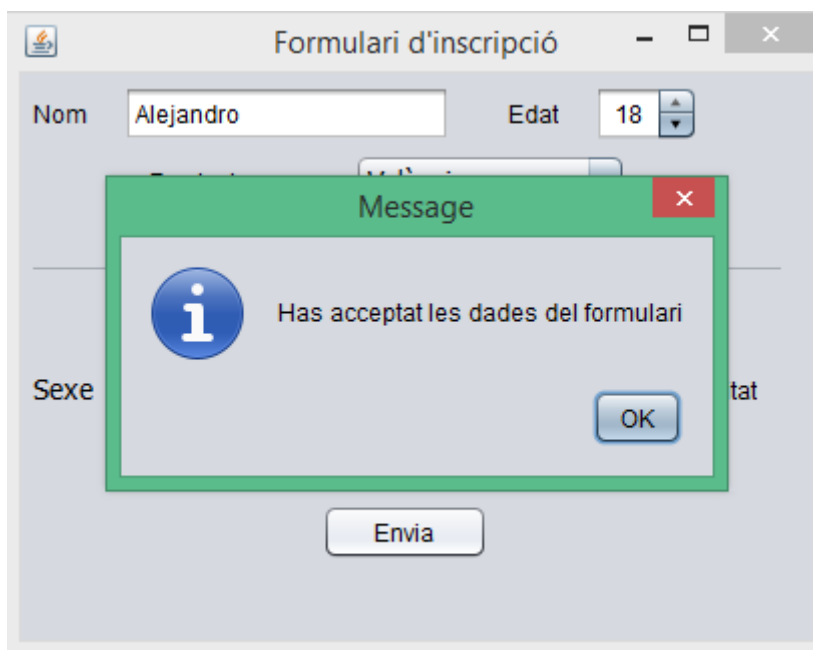
Per al camp JSpinner a partir del qual afegim la edat, has de establir un màxim de 99 anys i un mínim de 18. Fes una ullada a la teoria per a veure com es fa.

Estableix un JOptionPane, com el següent:





Si fem click en Yes apareixerà el següent diàleg:



A més, quan fem click en Ok, tornarà a la aplicació mostrant els valors per defecte.

Si fem click en No, eixirà completament de l'aplicació.

Si fem click en Cancel mostrarà novament el formulari.

Exercici 5

En aquest exercici treballarem amb els events. Aprofitant el anterior exercici veurem com podem anar actualitzant les dades a altres components.



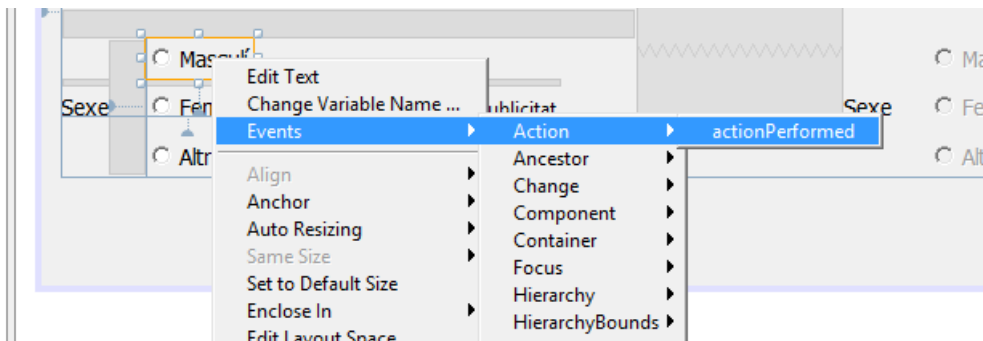
En primer lloc, fes un disseny com el següent:

Els camps de la dreta no poden ser modificats per l'usuari sinó que faran d'espill, configura-ho.

Abans que tot has de repetir l'acció del radio butons per a què no es marquen tots a la mateixa vegada.

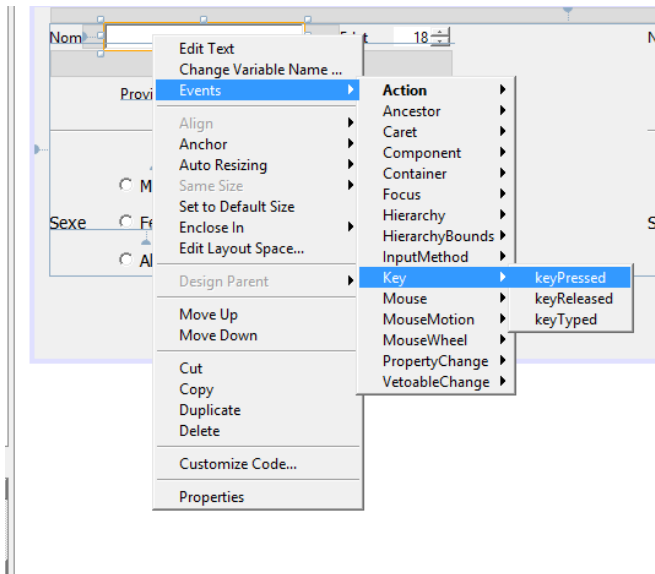
Ara pasem a la part dels events:

Els radio buttons tindran el event actionPerformed, que son events de butó. És a dir, quan fem click ocorre alguna cosa. Açò també serà el mateix per al checkbox que hem posat.



Per al nom, el event serà el de key pressed:





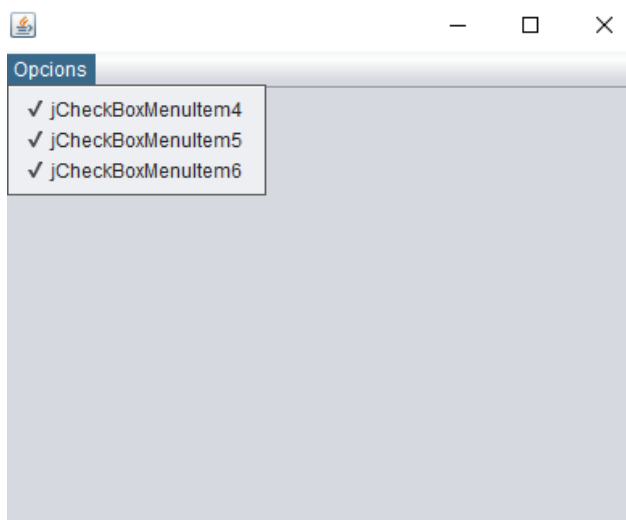
Per al Combo Box emplearem un event que s'anomena canvi de item.

Per al Spinner emprèn el event stateChange.

Comproba que la part de la dreta actua d'espill.

Exercici 6

A partir del formulari creat al exercici 4, crearem un menu amb una sèrie de items amb què podrem deshabilitar els camps del formulari. La forma es la següent:



Les opcions son checkbox, que depenen si estan marcades o no apareixen o desapareixen.



Exercici 7

Crea una finestra en la que jugues a encertar un numero. Per una banda, has d'escriure un numero y per altra es generarà un número aleatori en el moment que polses sobre un butó anomenat "juga".

Si no vols escriure el número, pots tindre l'opció de fer click dret i desplegar una opció que siga la de generar número aleatori en la casella que escriu el número el jugador. Aço ho farem amb un PopUp Menu.

