1.1.1.1 Diálogo showOptionDialog

Este tipo de diálogo é o máis persoalizable de todos. Con el podemos crear diálogos de tipo showMessageDialog e diálogos de tipo showConfirmDialog nos que é posible configurar calquera elemento do diálogo. O diálogo de tipo showOptionDialog emprégase para amosar un diálogo modal que contén unha mensaxe e un ou máis botóns, sendo posible detectar cal foi o botón premido polo usuario. Para xerar este tipo de diálogo emprégase a clase JOptionPane. Esta clase proporciónanos un único método, pero en función dos parámetros que lle pasemos, a xanela de diálogo variará o seu aspecto. O método que empregamos para crear unha xanela de diálogo de tipo showOptionDialog é o seguinte:

- public static int showOptionDialog(Component parentComponent, Object message,String title, int optionType, int messageType, Icon icon, Object [] options, Object initialValue)

De seguido explícase o significado de cada parámetro:

- parentComponent fai referencia a quen é a xanela pai (graficamente) do diálogo.
- message: mensaxe que se amosa na xanela
- title: título da xanela
- optionType: en función do valor indicado amosaranse distintas configuracións de botóns. Os valores posibles son os seguintes: `DEFAULT_OPTION`, `YES_NO_OPTION`, `YES_NO_CANCEL_OPTION`, `OK_CANCEL_OPTION` (Todas son constantes da clase JOptionPane). P.e., `YES_NO_CANCEL_OPTION` presentará tres botóns (botón `YES`, botón `NO` e botón `CANCEL`).
- messageType: en función do valor indicado amosaranse distintas iconas predefinidas xunto á mensaxe. Os valores posibles son os seguintes: `ERROR_MESSAGE`, `INFORMATION_MESSAGE`, `WARNING_MESSAGE`, `QUESTION_MESSAGE`, `PLAIN_MESSAGE` (Todas son constantes da clase JOptionPane).
- icon: se en lugar dunha icona predefinida queremos amosar unha persoalizada, o indicaremos mediante este parámetro. Se empregamos este parámetro, o valor dado ao parámetro messageType é ignorado.
- Options: mediante este parámetro defínense os textos dos botóns no caso de que queiramos empregar botóns persoalizados.
- initialValue: en caso de que empreguemos botóns persoalizados, mediante este parámetro podemos indicar cal é o botón preseleccionado por defecto.

O método devolve un valor enteiro que indica o botón elixido polo usuario o CLOSED_OPTION no caso de que o usuario pechase o diálogo sen premer ningún botón.

De seguido imos desenvolver unha pequena aplicación que amosa o emprego deste método para distintas configuracións de xanela. O seu aspecto

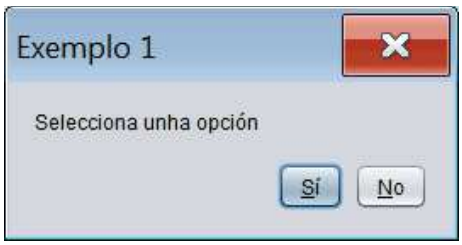
será o seguinte:

Al pulsar sobre o primeiro botón, xeraremos un showOptionDialog que amose dous botóns (sí e no). Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnSiNonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int  seleccion=JOptionPane.showOptionDialog(this,  "Selecciona  unha  opción",  "Exemplo  1",
JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE, null, null, null);
        String mensaxe="Xanela pechada polo usuario";
        switch(seleccion)
        {
            case JOptionPane.YES_OPTION:    mensaxe="Pulsado o botón Sí";
                                            break;
            case JOptionPane.NO_OPTION:     mensaxe="Pulsado o botón Non";
                                            break;
        }

        JOptionPane.showMessageDialog(this, mensaxe, "Resultado",JOptionPane.INFORMATION_MESSAGE);
    }
```

O resultado de premer o primeiro botón é o seguinte:



Fixémonos que o texto dos botóns está en español. Isto é debido a que cando indicamos a configuración de botóns que queremos (YES_NO_OPTION), o compilador facendo emprego do seu sistema de internacionalización aplica as cadeas correspondentes para a configuración actual da máquina virtual (configuración española). Outro tema destacable é ver como recollemos a acción realizada polo usuario sobre o diálogo. Neste caso o usuario pode facer unha das seguintes accións: pechar a xanela, premer o botón sí ou premer o botón no. O método showOptionDialog devolveranos un valor enteiro en función da acción realizada. Para cada botón predefinido, a clase JOptionPane ten unha constante co valor devolto ao premer nese botón. P.e., para o botón sí é JOptionPane.YES_OPTION e para o botón no é JOptionPane.NO_OPTION (en xeral, as constantes de JOptionPane de tipo TEXTOBOTON_OPTION representan valores devoltos polos botóns predefinidos da clase). No caso de que o valor devolto non sexa ningún deles significará que o usuario pechou a xanela.

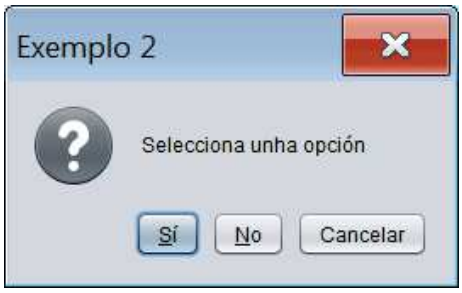
Ao premer sobre o segundo botón, xeraremos un showOptionDialog que amose tres botóns (sí, no e cancelar) e unha icona predefinida. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnSiNonCancelarIconaPredefinidaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int  seleccion=JOptionPane.showOptionDialog(this,  "Selecciona  unha  opción",  "Exemplo  2",
JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);
        String mensaxe="Xanela pechada polo usuario";
        switch(seleccion)
        {
            case JOptionPane.YES_OPTION:    mensaxe="Pulsado o botón Sí";
                                            break;
            case JOptionPane.NO_OPTION:     mensaxe="Pulsado o botón Non";
                                            break;
            case JOptionPane.CANCEL_OPTION: mensaxe="Pulsado o botón Cancelar";
                                            break;

        }

        JOptionPane.showMessageDialog(this, mensaxe, "Resultado",JOptionPane.INFORMATION_MESSAGE);
    }
```

O resultado de premer o segundo botón é o seguinte:



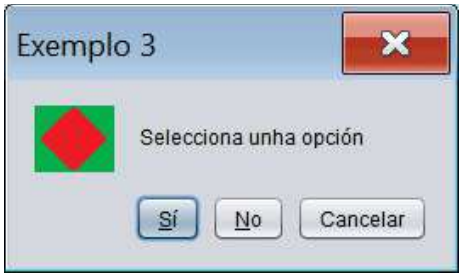
Ao premer sobre o terceiro botón, xeraremos un showOptionDialog que amose tres botóns (sí, no e cancelar) e unha icona persoalizada. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnSiNonCancelarIconaPersoalizadaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        ImageIcon icona=new ImageIcon(getClass().getResource("/imaxes/icona.png"));
        int  seleccion=JOptionPane.showOptionDialog(this,  "Selecciona  unha  opción",  "Exemplo  3",
JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, icona, null, null);
        String mensaxe="Xanela pechada polo usuario";
        switch(seleccion)
        {
            case JOptionPane.YES_OPTION:    mensaxe="Pulsado o botón Sí";
                                            break;
            case JOptionPane.NO_OPTION:     mensaxe="Pulsado o botón Non";
                                            break;
            case JOptionPane.CANCEL_OPTION: mensaxe="Pulsado o botón Cancelar";
                                            break;

        }

        JOptionPane.showMessageDialog(this, mensaxe, "Resultado",JOptionPane.INFORMATION_MESSAGE);
    }
```

O resultado de premer o terceiro botón é o seguinte:



Ao premer sobre o cuarto botón, xeraremos un showOptionDialog que amose tres botóns persoalizados. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnBotonsPersoalizadosActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String botons[]={"Chove","Non chove","Poidera ser"};
        int   seleccion=JOptionPane.showOptionDialog(this,   "¿Choverá   pola   tarde?",   "Exemplo   4",
JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null, botons, null);
        String mensaxe="Xanela pechada polo usuario";
        switch(seleccion)
        {
            case 0:    mensaxe="Vai chover";
                                        break;
            case 1:     mensaxe="Non vai chover";
                                        break;
            case 2: mensaxe="Poidera ser que chova. Xa veremos";
                                        break;

        }

        JOptionPane.showMessageDialog(this, mensaxe, "Resultado",JOptionPane.INFORMATION_MESSAGE);

    }
```

O resultado de premer o cuarto botón é o seguinte:

Para persoalizar os textos dos botóns creamos un arrai de Strings. Cada elemento deste arrai mapearase sobre un botón do diálogo. A orde dos botóns no diálogo será a orde dos Strings no arrai de Strings. Para recuperar o botón premido farémolo pola súa posición dentro do arrai de Strings, de xeito que se prememos o primeiro botón, o diálogo devolverá un 0, se prememos sobre o segundo botón o diálogo devolverá un 1, etc.

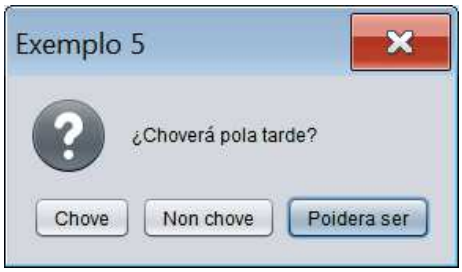
Por último, ao premer sobre o quinto botón, xeraremos un showOptionDialog que amose tres botóns persoalizados e ademais preseleccione un deles. Por defecto se non se indica nada sempre preseleccionarase o primeiro botón. A preselección persoalizada de botóns unicamente pódese aplicar cando os botóns son persoalizados. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private  void  btnBotonsPersoalizadosEValorPreseleccionadoActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        String botons[]={"Chove","Non chove","Poidera ser"};
        int   seleccion=JOptionPane.showOptionDialog(this,   "¿Choverá   pola   tarde?",   "Exemplo   4",
JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null, botons, botons[2]);
        String mensaxe="Xanela pechada polo usuario";
        switch(seleccion)
        {
            case 0:    mensaxe="Vai chover";
                                        break;
            case 1:     mensaxe="Non vai chover";
                                        break;
            case 2: mensaxe="Poidera ser que chova. Xa veremos";
                                        break;

        }

        JOptionPane.showMessageDialog(this, mensaxe, "Resultado",JOptionPane.INFORMATION_MESSAGE);
    }
```

O resultado de premer o quinto botón é o seguinte:



Como se pode observar na imaxe anterior, o terceiro botón está preseleccionado. Para facelo indicamos no último parámetro do método cal de tódolos elementos do arrai de Strings que empregamos para xerar os botóns é o que queremos preseleccionar.

1.1.1.2 Consideracións finais

Se prestamos atención aos construtores dalgúns dos diálogos predefinidos que acabamos de explicar, veremos que en algunhas ocasións algúns dos parámetros destes construtores son de clase Object. Non obstante, nos, para eses parámetros, empregamos obxectos de clase String, xa que habitualmente nunha xanela de diálogo amósanse cadeas de texto ao usuario. Se para eses parámetros en lugar de empregar obxectos String empregamos obxectos doutra clase, amosarase a información que devolvan os seus métodos toString. Se ademais os obxectos empregados son dalgunha clase de compoñente gráfico podemos obter resultados curiosos. P.e.: para as seguintes liñas de código:

```
JTextArea txtarInfo=new JTextArea("Esta información\namósase nun JTextArea");
JOptionPane.showMessageDialog(this, txtarInfo);
```

amósase a seguinte xanela:

O máis habitual é traballar con Strings, pero é bo lembrar que sempre hai outras opcións