1.1.1 Diálogos predefinidos

Con anterioridade vimos como desenvolver xanelas de diálogo. Os diálogos creados eran xanelas cuxos compoñentes eran establecidos polo programador co fin de cumprir cos requisitos das aplicacións. Non obstante, a linguaxe java proporciónanos unha serie de xanelas de diálogo cun formato predefinido que se adecúan as situacións máis habituais. Unha destas xanelas de diálogo predefinidas témola empregado con gran asiduidade ao longo do desenvolvemento das diferentes actividades realizadas ata agora. Estamos a falar da seguinte xanela:

A xanela cuxo título é Mensaje (tamén veremos como persoalizar este texto para que sexa amosado en galego) é un diálogo modal que se xera escribindo a seguinte liña de código:

```
JOptionPane.showMessageDialog(this, "O botón 1 foi premido");
```

Esta liña afórranos o ter que crear unha xanela de diálogo, deseñar os seus compoñentes e xestionar os seus eventos. Simplifica a tarefa ao invocar un método da clase JOptionPane pasándolle a mensaxe que queremos amosarlle ao usuario.

A continuación imos estudiar os diferentes diálogos predefinidos proporcionados pola linguaxe java e como facer emprego deles.

1.1.1.1 Diálogo showMessageDialog

Este tipo de diálogo emprégase para amosar un diálogo modal que vai conter unha mensaxe. Para xerar este tipo de diálogo emprégase a clase JOptionPane. Esta clase proporciónanos tres métodos diferentes para crear diálogos de tipo showMessageDialog. En función do método emprgado para crear a xanela de diálogo o seu aspecto variará. Os métodos que podemos empregar para crear unha xanela de diálogo de tipo showMessageDialog son os seguintes:

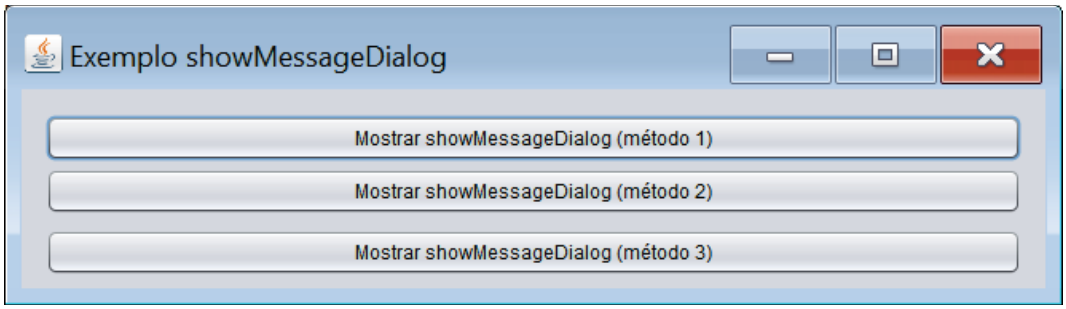
- public static void showMessageDialog(Component parentComponent, Object message)
- public static void showMessageDialog(Component parentComponent, Object message, String title, int messageType)

- public static void showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)

De seguido explícase o significado de cada parámetro:

- parentComponent fai referencia a quen é a xanela pai (graficamente) do diálogo.
- message: mensaxe que se amosara na xanela.
- title: título da xanela.
- 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, indicarémolo mediante este parámetro. Se empregamos este parámetro o valor dado ao parámetro messageType é ignorado.

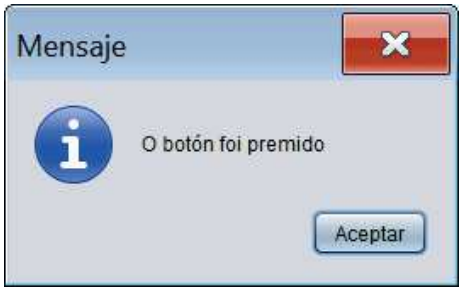
A continuación imos desenvolver unha pequena aplicación que amosa o emprego de cada un dos métodos que acabamos de comentar. O seu aspecto será o seguinte:



Ao premer sobre o primeiro botón, xeraremos un showMessageDialog empregando o primeiro método. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnMetodo1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        JOptionPane.showMessageDialog(this, "O botón foi premido");
    }
```

O resultado de premer o primeiro botón é o seguinte:

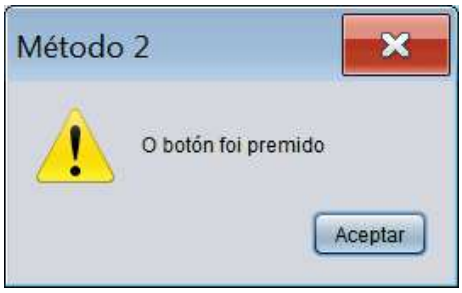


Tal e como era de esperar unicamente persoalizamos a mensaxe da xanela. Fixémonos que o título da xanela ten un texto predefinido (dependente dos locais da nosa máquina virtual, neste caso configuración española).

Ao premer sobre o segundo botón, xeraremos un showMessageDialog empregando o segundo método. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnMetodo2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        JOptionPane.showMessageDialog(this,     "O     botón     foi     premido",     "Método     2",
JOptionPane.WARNING_MESSAGE);
    }
```

O resultado de premer no segundo botón é o seguinte:

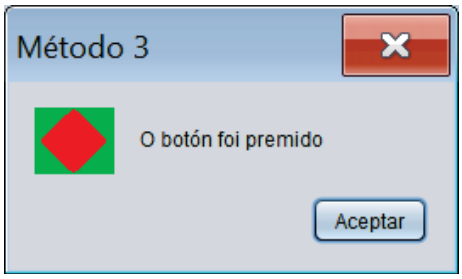


Neste caso, ademais de persoalizar a mensaxe da xanela, tamén persoalizamos o título da xanela e a icona.

Por ultimo, ao premer sobre o terceiro botón, xeraremos un showMessageDialog empregando o terceiro método. Para elo, na implementación do actionPerformed do botón escribimos o seguinte código:

```
    private void btnMetodo3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        ImageIcon icona=new ImageIcon(getClass().getResource("/imaxes/icona.png"));
        JOptionPane.showMessageDialog(this,       "O       botón       foi       premido",       "Método
3",JOptionPane.WARNING_MESSAGE, icona);
    }
```

O resultado de premer o terceiro botón é o seguinte:



Neste caso, persoalizamos a mensaxe da xanela, o título da xanela e a icona, pero coa diferencia de que esta vez a icona non é unha entre as que nos proporciona o sistema senón que é unha icona persoalizada.