Ventanas de entrada de datos, mensajes y archivos

Introducción a las ventanas de diálogo. Pepe Calo.

2. Ventanas de diálogo con Swing

- **JFC (*Java Foundation Classes*)** es un conjunto de funciones/clases para **crear interfaces gráficas de usuario (GUI)** y añadir funcionalidad gráfica e interactividad a las aplicaciones Java que estudiaréis en la materia de ***Desenvolvemento de Interfaces***.

- Incluye: **componentes gráficos de Swing** (botones, paneles, tablas, ventanas, etc;), **configuración de apariencia, API Java 2D, API de accesibilidad** (lectores de pantalla, Braille,…); **internacionalización** (gestión de idiomas del mundo,..)

2. Ventanas de diálogo (II)

- En la materia de **Acceso a Datos** vamos a dar una **pequeña introducción a dos componentes de diálogo** que nos facilitarán la introducción de datos, mostrar mensajes o selección de ficheros hasta que lo estudiéis en otras materias:

  - ***JOptionPane:***
    - https://docs.oracle.com/en/java/javase/20/docs/api/java.desktop/javax/swing/JOptionPane.html
  - ***JFileChooser:***
    - https://docs.oracle.com/en/java/javase/20/docs/api/java.desktop/javax/swing/JFileChooser.html

2. Ventanas de diálogo (III)

- Una ventana de diálogo es una **subventana independiente** destinada a llevar un aviso temporal además de la ventana principal de la aplicación Swing.

- Suelen usarse para **mostrar mensajes de error o una advertencia**, presentar imágenes, árboles de directorios, etc.

- Para facilitar el trabajo, existen la clases de utilidad:
  - ***JOptionPane*** para crear ventanas de diálogo estándar y sencillos.
  - Barras de progreso: ***JProgressBar*** *(ProgressMonitor)*
  - Elección de color: ***JColorChooser***.
  - Selección de archivos: ***JFileChooser***.
  - Diálogos de impresión: API de impresión.

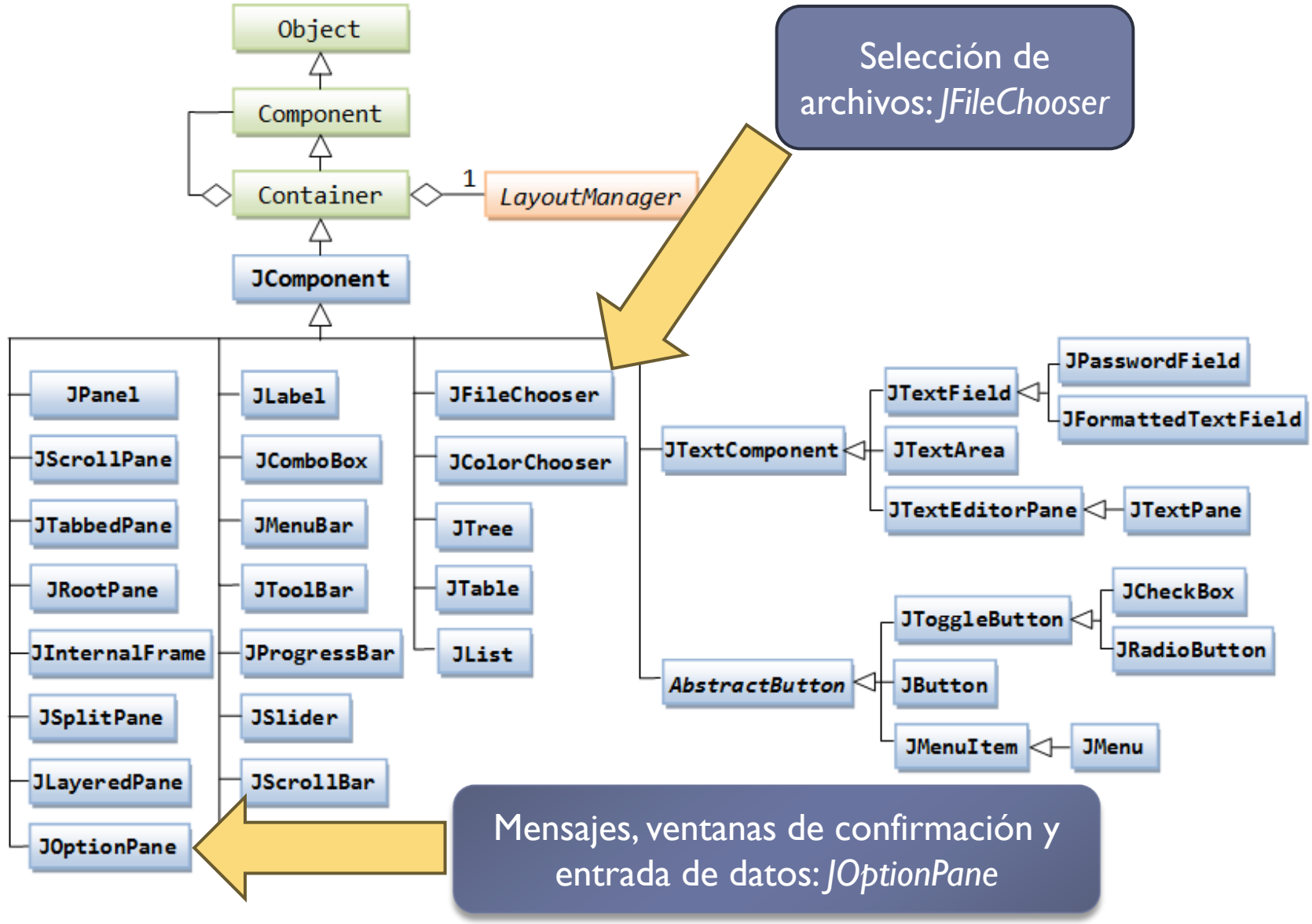
2. Ventanas de diálogo (IV)

- Paquetes principales del API:
  - ***javax.swing***
  - *javax.swing.event*

- Además de:

javax.accessibility
javax.swing
javax.swing.border
javax.swing.colorchooser
javax.swing.event
javax.swing.filechooser
javax.swing.plaf
javax.swing.plaf.basic
javax.swing.plaf.metal
javax.swing.plaf.multi
javax.swing.plaf.synth
javax.swing.table
javax.swing.text
javax.swing.text.html
javax.swing.text.html.parser
javax.swing.text.rtf
javax.swing.tree
javax.swing.undo

2. Componentes Swing (*JComponent*)



01.00.01 ***JOptionPane***
Ventanas de diálogo simples

Pepe Calo

1. JOptionPane (I)

- ***JOptionPane*** permite crear y personalizar rápidamente varios tipos diferentes de cuadros de diálogo.

- Proporciona soporte para diseñar **cuadros de diálogo estándar**, con iconos, especificar el título y el texto del cuadro de diálogo y personalizar el texto del botón.

- La compatibilidad con iconos de le **permite especificar fácilmente qué icono muestra el cuadro de diálogo.**

- **Son modales**, esto es, bloquea el acceso a la ventana padre.

1. JOptionPane (II): iconos

- Puede utilizar un **icono personalizado**, ningún icono o cualquiera de los **cuatro iconos estándar**:

- Cada apariencia tiene sus propias versiones de los cuatro íconos estándar.

1. JOptionPane (III)

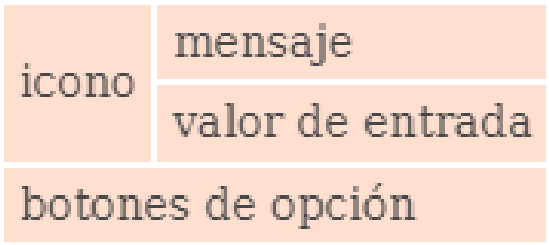
- La forma más sencilla de crear y mostrar diálogos con ***JOptionPane*** es por medio de los métodos ***showXxxDialog***:

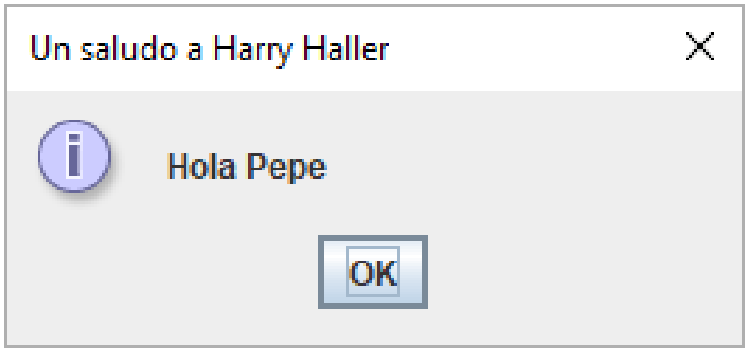
Método	Descripción
showConfirmDialog	Pregunta de confirmación, como sí/no/cancelar.
showInputDialog	Solicita entrada de datos.
showMessageDialog	Muestra un mensaje informativo
showOptionDialog	Permite personalizar el *JOptionPane*.

Existe una versión para marcos internos, ***JInternalFrame*** *con métodos de la forma:* ***showInternalXxxx***

1. JOptionPane (IV)

- El formato del mensaje tiene una apariencia siguiendo la siguiente estructura:





1. JOptionPane (V): parámetros

- Los parámetros de los método showXxxDialog son los siguientes:
  - ***parentComponent***: componente **padre**, el JFrame. Si el valor es ***null***, se usa el JFrame por defecto y se centra en la pantalla.
  - ***mensaje***: **mensaje** de la ventana de diálogo. **Normalmente *String***, pero puede ser cualquier objeto:
    - ***Object[]***: será interpretado como una serie de mensajes (uno por objeto) situados en vertical.
    - ***Component***: mostrará el componente en la ventana de diálogo.
    - ***Icon***: el icono será mostrado dentro de un JLabel.
    - ***Otros***: se convierten a String llamando al método toString() y mostrándolo dentro de un JLabel.
  - ***Título***: título de la ventana.

1. JOptionPane (V): parámetros (II)

- ***messageType***: define el estilo del mensaje. Los posibles valores son:
  - ERROR_MESSAGE
  - INFORMATION_MESSAGE
  - WARNING_MESSAGE
  - QUESTION_MESSAGE
  - PLAIN_MESSAGE
- ***optionType***: conjunto **botones** de opción que aparen debajo de la ventana. Se pueden proporcionar otro botones usando el parámetro *options*.
  - DEFAULT_OPTION
  - YES_NO_OPTION
  - YES_NO_CANCEL_OPTION
  - OK_CANCEL_OPTION

1. JOptionPane (V): parámetros (III)

- ***options***: es una descripción más detallada del **conjunto de botones** que aparecen en la parte inferior de la ventana. Lo usual es un array de String, pero puede ser un array de Object:
  - ***Component***: el componente se añade a la lista de botones.
  - ***Icon***: se crea un JButton con este icono.
  - ***Otros***: el objeto se convierte en String (toString()) y se emplea como etiqueta del botón.
- ***Icono***: **icono** de la ventana de diálogo. El valor por defecto está determinado por el tipo de mensaje.
- ***initialValue***: **valor por defecto** para ventanas de tipo *input*.

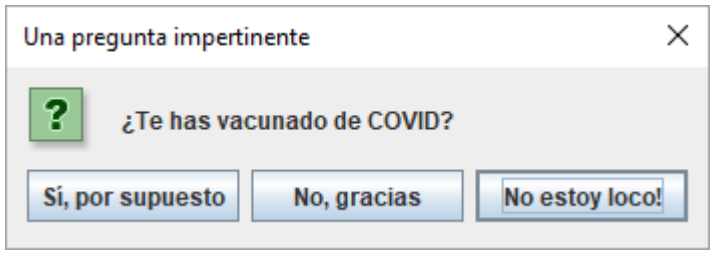
1. JOptionPane: valor devuelto.

- Los métodos ***showXxxDialog*** devuelven un **entero**, cuyos valores posibles son las constantes que referencian al botón pulsado:

  - YES_OPTION
  - NO_OPTION
  - CANCEL_OPTION
  - OK_OPTION
  - CLOSED_OPTION

1. JOptionPane. Ejemplo

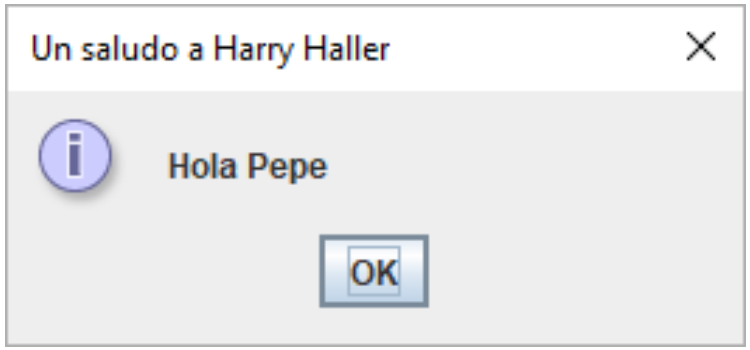
```
Object[] opcionesBoton = {"Sí, por supuesto", "No, gracias", "No estoy loco!"};
int resultado = JOptionPane.showOptionDialog(this,
    "¿Te has vacunado de COVID?", // mensaje
    "Una pregunta impertinente", // título
    JOptionPane.YES_NO_CANCEL_OPTION, // OptionType
    JOptionPane.QUESTION_MESSAGE, // Tipo de mensaje
    null, // icono
    opcionesBoton,
    opcionesBoton[2]); // valor por defecto
```



2. JOptionPane: ***showMessageDialog***

- ***showMessageDialog*** : muestra un mensaje simple con un botón. (***this*** *es la referencia al formulario)*

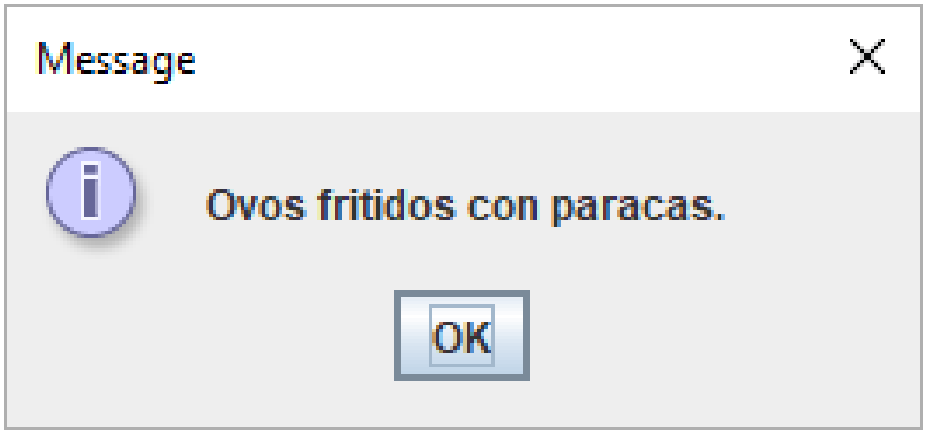
```
JOptionPane.showMessageDialog(this, "Hola Pepe",
                    "Un saludo a Harry Haller",
                    JOptionPane.INFORMATION_MESSAGE);
```



2. JOptionPane: ***showMessageDialog***

- Con título e icono por defecto:

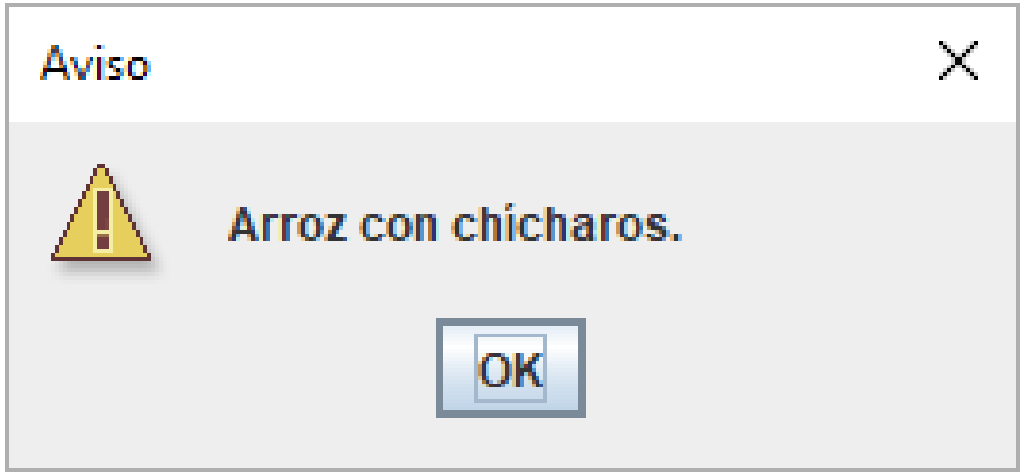
  *JOptionPane.**showMessageDialog**(this, "Ovos fritidos con paracas.");*



2. JOptionPane: ***showMessageDialog***

- Con título, icono de aviso:

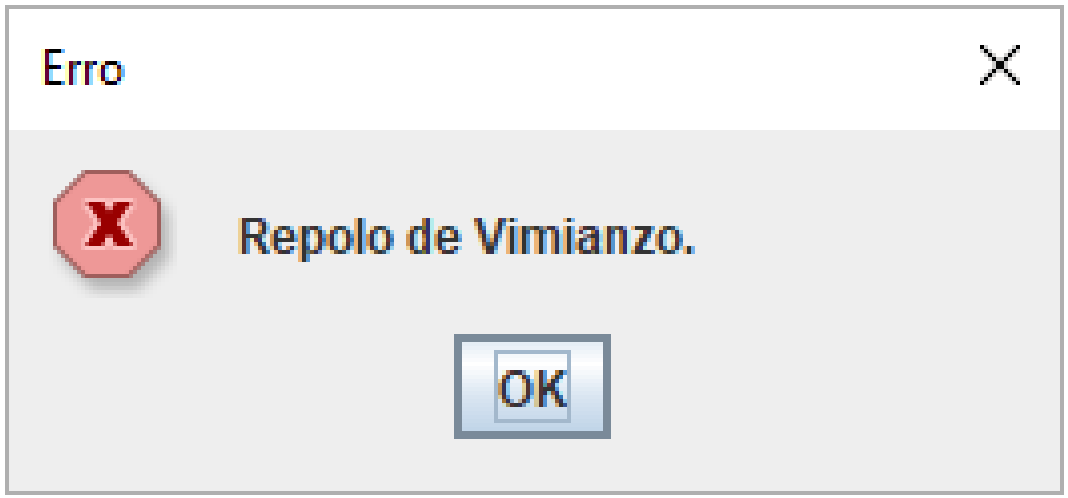
```
JOptionPane.showMessageDialog(this,
                "Arroz con chícharos.", "Aviso",
                    JOptionPane.WARNING_MESSAGE);
```



2. JOptionPane: ***showMessageDialog***

- Con título, icono de error:

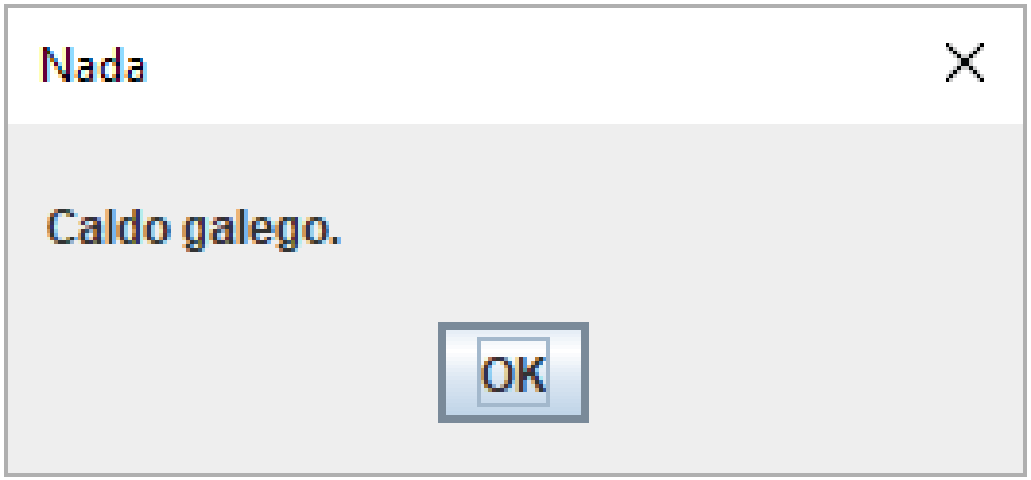
```
JOptionPane.showMessageDialog(this,
                "Repolo de Vimianzo.", "Erro",
                    JOptionPane.ERROR_MESSAGE);
```



2. JOptionPane: ***showMessageDialog***

- Con título, sin icono:

```
JOptionPane.showMessageDialog(this,
                "Caldo galego.", "Nada",
                    JOptionPane.PLAIN_MESSAGE);
```



2. JOptionPane: ***showMessageDialog***

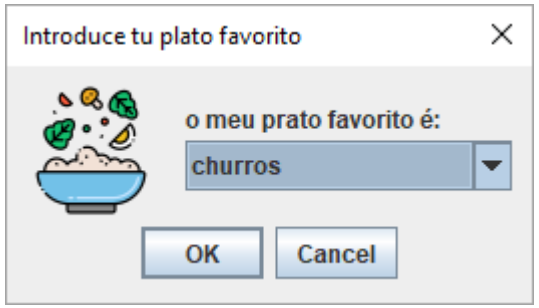
- Con título, icono personalizado:

```
ImageIcon icoSalada
        = new ImageIcon( getClass().getResource("/images/ensalada.png"));
JOptionPane.showMessageDialog(this, "Salada con verde.",
        "Primeiro prato", JOptionPane.INFORMATION_MESSAGE,
        iconaSalada);
```

2. JOptionPane: ***showInputDialog***

- ***showInputDialog*** : es el único método ***showXxxDialog*** que no devuelve un entero.
- **Devuelve un objeto**, normalmente un *String*:

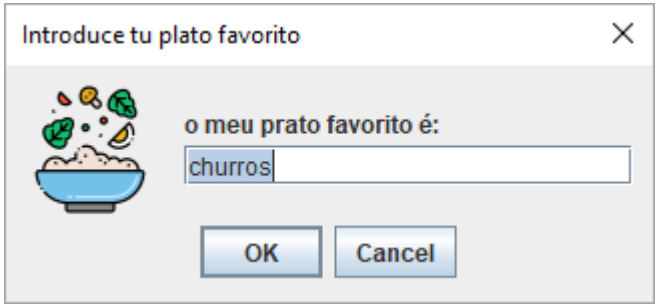
```
Object[] platoFavorito = {"chícharos", "doce", "churros"};
String s = (String)JOptionPane.showInputDialog(this,
    "o meu prato favorito é: ",
    "Introduce tu plato favorito",
    JOptionPane.PLAIN_MESSAGE,
    iconaSalada,
    platoFavorito,
    "churros");
```



2. JOptionPane: ***showInputDialog*** (II)

- Si ponemos *null* en el array de opciones aparecerá una caja de texto:
- **Devuelve un objeto**, normalmente un *String*:

```
String s = (String)JOptionPane.showInputDialog(this,
    "o meu prato favorito é: ",
    "Introduce tu plato favorito",
    JOptionPane.PLAIN_MESSAGE,
    iconaSalada,
    null,
    "churros");
```



01.00.02 Selectores de archivo *JFileChooser*

Pepe Calo

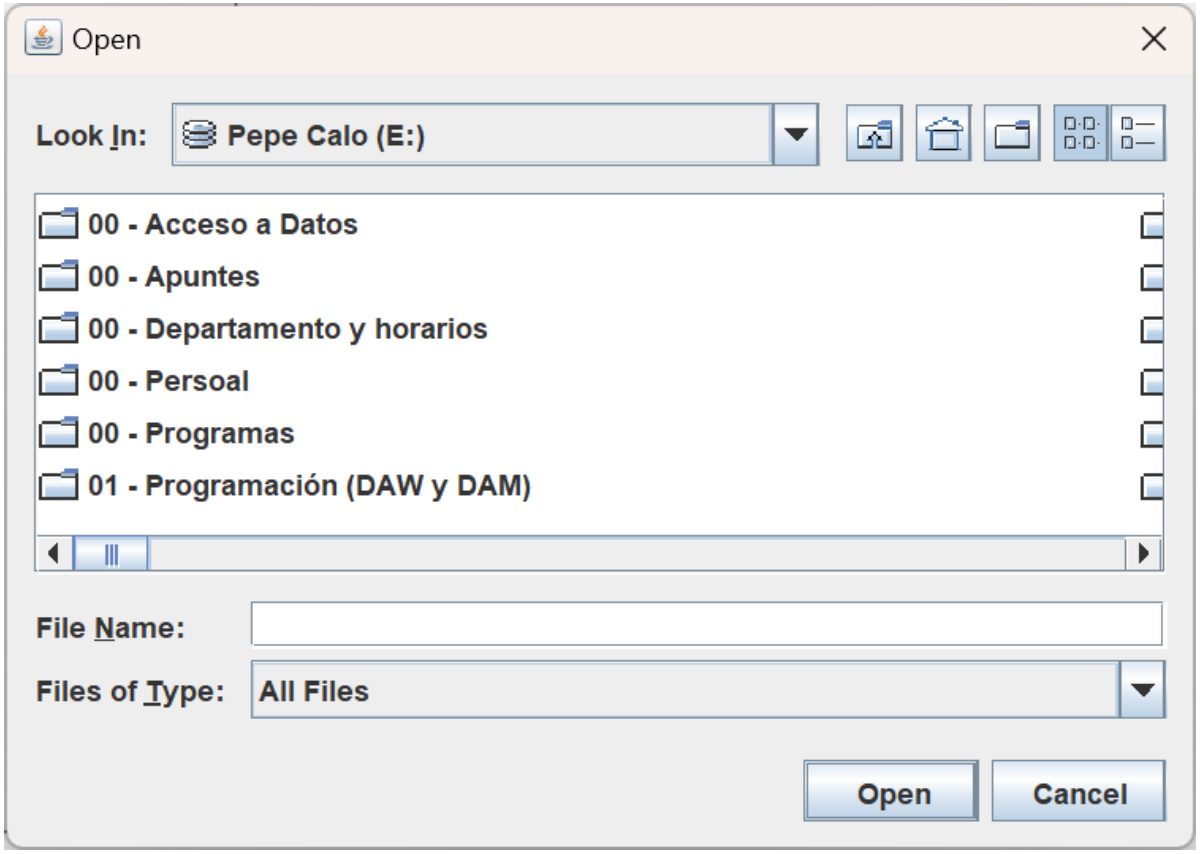
JFileChooser

- Los selectores de archivos **proporcionan una GUI para navegar por el sistema de archivos y luego elegir un archivo o directorio** de una lista, o introducir el nombre de un archivo o directorio.

- Normalmente usa la clase **JFileChooser** para mostrar un cuadro de diálogo modal que contiene el selector de archivos. *Otra forma de presentar un selector de archivos es agregar una instancia de JFileChooser a un contenedor (ventana etc)*

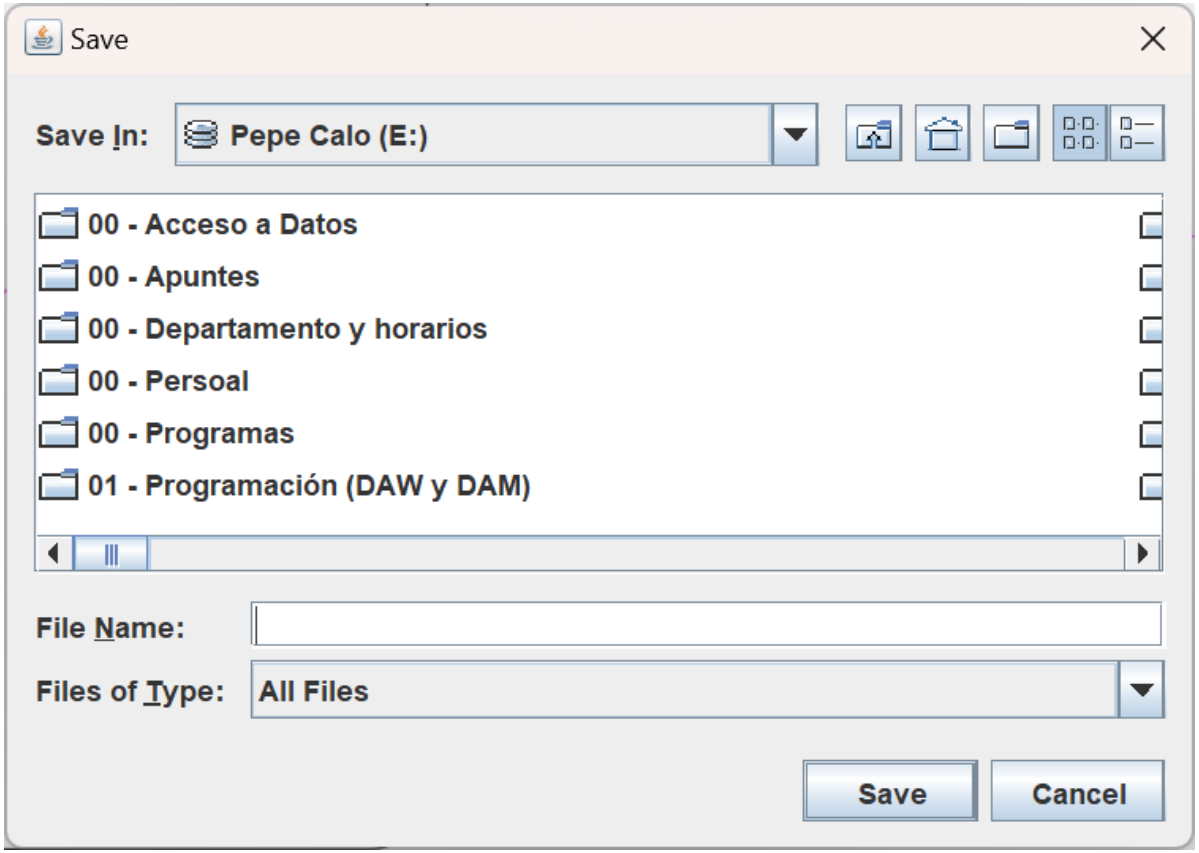
JFileChooser (II)

- Ejemplo selección/apertura de archivo: ***showOpenDialog***



JFileChooser (III)

- Ejemplo para guardar de archivo: ***showSaveDialog***



JFileChooser: ejemplo (con filtro)

```
JFileChooser fclmaxe = new JFileChooser();
 FileNameExtensionFilter filtro
        = new FileNameExtensionFilter(
                        "Imágenes JPG y PNG", "jpg", "png");
   fclmaxe.setFileFilter(filtro);
   int valorSel = fclmaxe.showOpenDialog(null);
   if(valorSel == JFileChooser.APPROVE_OPTION) {
      System.out.println("Has seleccionado la imagen: " +
         fclmaxe.getSelectedFile().getName());
   }
```

JFileChooser: directorio actual.

- ***showOpenDialog*** recoge el componente padre de la ventana de diálogo y afecta a la posición de la ventana de diálogo.

- Por defecto muestra los archivos del directorio de trabajo del usuario, pero puede especificarse el directorio inicial de varios modos:

  - En el constructor:
    - ***JFileChooser*** *fc =* ***new*** *JFileChooser("e:\\");*

  - Por medio del método:
    - *fc.****setCurrentDirectory****(new File("e:\\");*

JFileChooser: selección de archivo/dir

- ***showOpenDialog/showSaveDialog*** recoge devuelven un entero que indica si se ha seleccionado un archivo: **APPROVE_OPTION** o **CANCEL_OPTION**

- Una vez seleccionado un archivo o directorio (en ese caso debe indicarse que se permite selección de directorios) puede invocarse al método ***getSelectedFile()*** para recuperar el archivo *(File)*:

  - ***File*** *arquivo* = *fcImaxe.getSelectedFile();*

JFileChooser: selección de arch/dir (II)

- Una vez recuperado el archivo podemos obtener muchos datos del mismo (lo veremos en la unidad de archivos):

  - ***File*** *arquivo* = *fcImaxe.getSelectedFile();*
  - *arquivo.getPath();*
  - *arquivo.getName();*
  - *arquivo.isDirectory();*
  - *arquivo.exists();*
  - … *arquivo.delete();*
  - …

JFileChooser: selección de arch/dir (III)

- Se puede utilizar la misma instancia de la JFileChooser para mostrar un cuadro de diálogo estándar para guardar.
  - *int valor = fc.**showSaveDialog**(null);*
- Al utilizar la misma instancia del JFileChooser:
  - Recuerda el directorio actual entre usos, por lo que las versiones para abrir y guardar comparten automáticamente el mismo directorio actual.
  - Sólo se personalizar un selector de archivos, y las personalizaciones se aplican tanto a la versión para abrir como para guardar.

JFileChooser: selección de arch/dir (IV)

- Se puede cambiar el modo de selección de archivos, por ejemplo para seleccionar directorios:
  - *fc.**setFileSelectionMode**(JFileChooser.DIRECTORIES_ONLY);*
- Además, existen ostro modos de selección:
  - *FILES_AND_DIRECTORIES*.
  - *FILES_ONLY* (por defecto).

JFileChooser: Filtros

- Por defecto, el JFileChooser **muestra todos los archivos y directorios, excepto los ocultos**.

- Pueden programarse filtros de archivos para escoger algún tipo de archivo o directorio.

- El JFileChooser llama al método ***accept*** (de *FileFilter*) determina qué se mostrará

JFileChooser: tipos de filtros

- Existen varios tipos de filtros:
  - ***setFileHidingEnabled****(false)*: para mostrar los archivos ocultos.
  - Subclases de la clase abstracta *FileFilter*: ***FileNameExtensionFilter***:

https://docs.oracle.com/en/java/javase/20/docs/api/java.desktop/javax/swing/filechooser/FileNameExtensionFilter.html

```
FileFilter filtro
    = new FileNameExtensionFilter("archivo JPEG", "jpg", "jpeg");
JFileChooser fc = ...;
fc.setFileFilter(filtro);
// fc.addChoosableFileFilter(filtro); // agrega a los seleccionables.
// fc. setAcceptAllFileFilterUsed(false);
```