Cuadros de dialogo

La clase ***JDialog*** es la clase que implementa cuadros de diálogo en ***swing***. Generalmente, dependen de una ventana principal (***JFrame***). Las ventanas y cuadros de diálogo pueden ser modales o no modales.

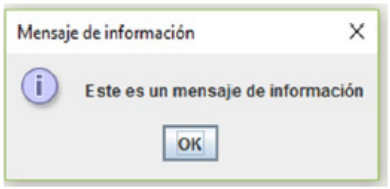
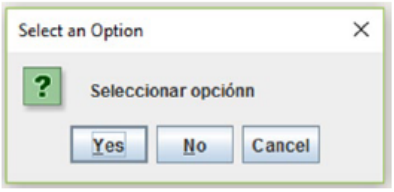
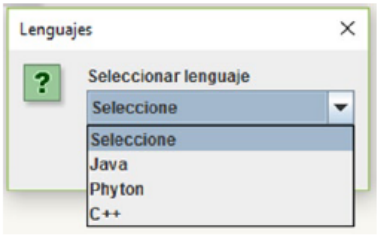
- **Cuadros de diálogo modales**: cuando un cuadro de diálogo modal es visible bloquea la entrada del usuario a todas las demás ventanas del programa. La clase ***JOptionPane*** crea ***JDialogs*** que son modales.
- **Cuadros de diálogo no modales**: permite el acceso a otras ventanas de la aplicación sin cerrar el cuadro de diálogo.

En el código, el tercer parámetro de la definición del cuadro de diálogo, es `boolean` e identifica si es un cuadro de diálogo modal o no.

```
JDialog miDiálogo = new JDialog(JFrame, tituloCuadroDiálogo, esModal);
```

La clase `JOptionPane` permite crear nuevos cuadros de diálogo o utilizar los más comunes. Los diferentes tipos de cuadros de diálogo se presentan en la tabla siguiente:

Tipos de cuadros de diálogo

Tipo/descripción	Código	Figura
Mensaje: para informar al usuario sobre algún hecho relevante. Pueden ser mensajes de información, advertencia, error o mensaje plano.	*JOptionPane.showMessage(JFrame, mensaje, tituloCuadro, tipo)* *Tipos:* • *JOptionPane.INFORMATION_MESSAGE* • *JOptionPane.WARNING_MESSAGE* • *JOptionPane.ERROR_MESSAGE* • *JOptionPane.PLAIN_MESSAGE*	
Confirmación: para realizar una pregunta al usuario con las posibles respuestas: sí, no o cancelar.	*JOptionPane.showConfirmDialog(JFrame, mensaje, tituloCuadro, tipo)* *Tipos:* • *JOptionPane.YES_NO_OPTION* • *JOptionPane.YES_NO_CANCEL_OPTION*	
Entrada: para solicitar una entrada del usuario.	*JOptionPane.showInputDialog(JFrame, mensaje)*	

Objetivo de aprendizaje

Al finalizar este ejercicio, tendrás la capacidad para generar diferentes cuadros de diálogo que faciliten la interacción de usuario con el programa.

Enunciado: Nómina

Se desea desarrollar un programa utilizando una interfaz gráfica de usuario que calcule la nómina de empleados de una empresa.

Para ello, el programa debe contar con una barra de menús con los siguientes ítems:

- Agregar empleado: genera una ventana donde se deben ingresar los datos de un empleado:
  - Nombre de tipo `String`.
  - Apellidos de tipo `String`.
  - Cargo, el cual puede ser directivo, estratégico u operativo, se puede implementar con un ***`JList`***.
  - Género, el cual puede ser masculino o femenino, se puede implementar con un ***`JCheckBox`***.
  - Salario por día de tipo `double`.
  - Días trabajados al mes desde 1 a 31, se puede implementar con un ***`JSpinner`***.
  - Otros ingresos de tipo `double`.
  - Pagos por salud de tipo `double`.
  - Aporte pensiones de tipo `double`.

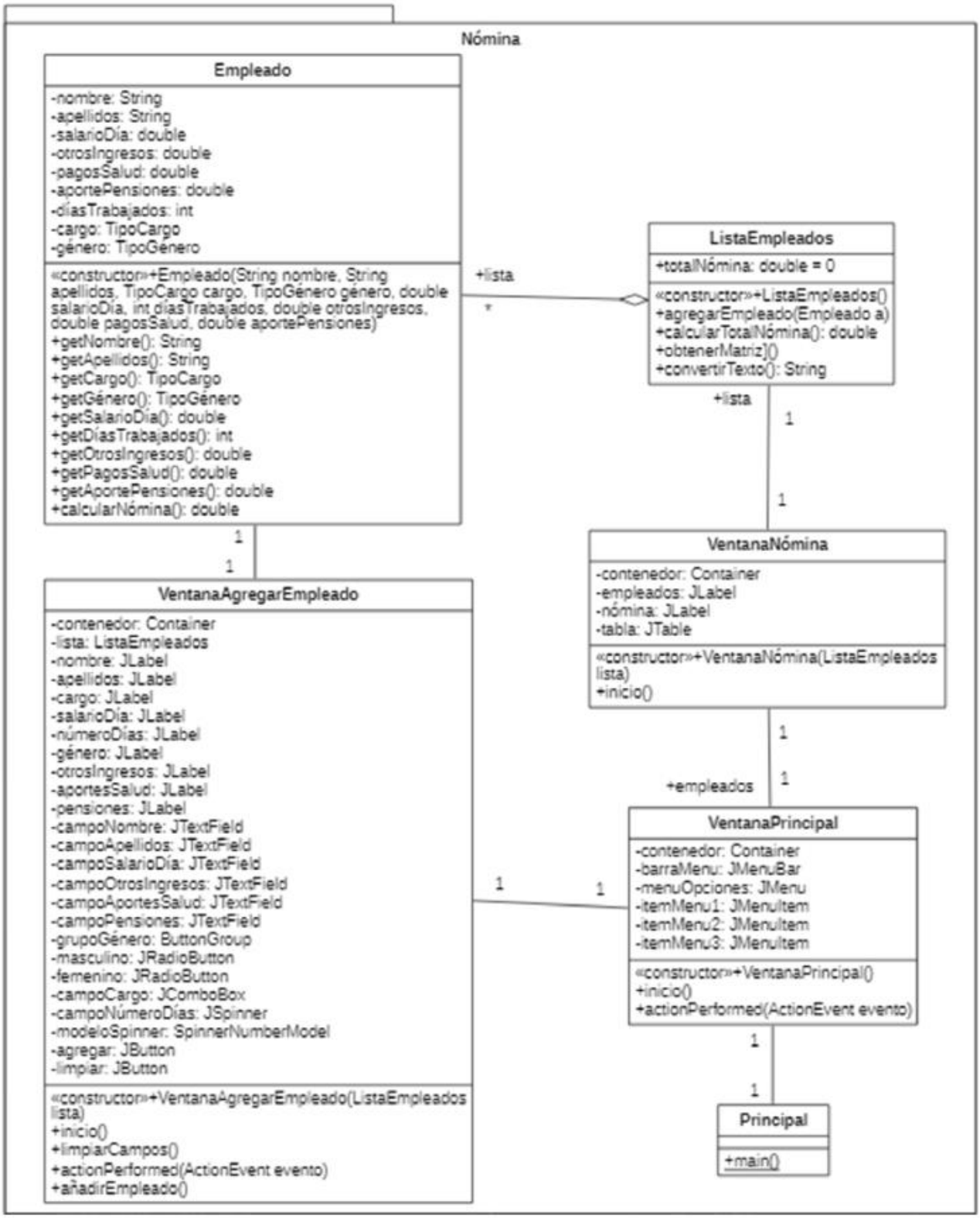
- Calcular nómina: genera una ventana donde se muestra en formato tabla los datos de los empleados ingresados. Cada fila de la tabla corresponde a un empleado. En las columnas se mostrarán los nombres, apellidos y sueldo de cada empleado. El sueldo de cada empleado se calcula como:

```
Salario mensual = (días trabajados * sueldo por día) + otros ingresos - pagos por salud - aporte pensiones
```

  En la parte inferior de la ventana, se calcula el total de la nómina de la empresa, calculada como la suma de los salarios mensuales de cada empleado.

- Guardar archivo: solicita una carpeta donde se va a generar un archivo de texto denominado "*Nómina.txt*" con los datos ingresados por cada empleado, su sueldo mensual calculado y el total de la nómina de la empresa.

Diagrama de clases



Explicación del diagrama de clases

Se ha definido un paquete denominado “Nómina” que incluye un conjunto de clases. El punto de entrada al programa es la clase Principal que cuenta con el método *main*. La clase Principal está relacionada mediante una asociación con la clase VentanaPrincipal que posee atributos privados para crear una ventana gráfica que cuenta con un contenedor de componentes gráficos (*Container*), una barra de menú (*JMenuBar*), una opción de menú (*JMenu*) y tres opciones de menú (*JMenuItem*). La clase VentanaPrincipal cuenta con un constructor y métodos para generar la ventana gráfica con sus componentes (inicio) y para gestionar los diferentes eventos surgidos al interactuar con esta ventana (*actionPerformed*).

La clase VentanaPrincipal está conectada por medio de una relación de asociación con las clases VentanaAgregarEmpleado y VentanaNomina. Las multiplicidades de estas relaciones son uno en cada extremo, lo que indica que se creará una sola ventana de su tipo cuando se invoquen desde la ventana principal.

En primer lugar, la clase VentanaAgregarEmpleado define una ventana que permite ingresar los datos de un empleado a agregar. Para ello, la clase VentanaAgregarEmpleado cuenta con atributos para definir los diferentes componentes gráficos de la ventana: un contenedor de componentes gráficos (*Container*); etiquetas para identificar los datos a ingresar (*JLabel*); campos de texto para ingresar los datos del empleado (*JTextField*); botones de radio (*JRadioButton*) agrupados (*ButtonGroup*) para identificar el género de empleado; una lista (*JComboBox*) para identificar el cargo del empleado, un selector numérico para ingresar la cantidad de días trabajados (*JSpinner* y *SpinnerNumberModel*) y botones para agregar un empleado o borrar un empleado (*JButton*). La clase VentanaAgregarEmpleado tiene un constructor y métodos para generar la ventana con sus componentes (inicio); gestionar los diferentes eventos surgidos al interactuar con esta ventana (*actionPerformed*); agregar un empleado (añadirEmpleado) y limpiar los campos de ingreso de datos (limpiarCampos).

La clase VentanaAgregarEmpleado se vincula con la clase Empleado por medio de una relación de asociación que tiene una multiplicidad de uno en cada extremo, lo que significa que la ventana creará un único empleado a la vez.

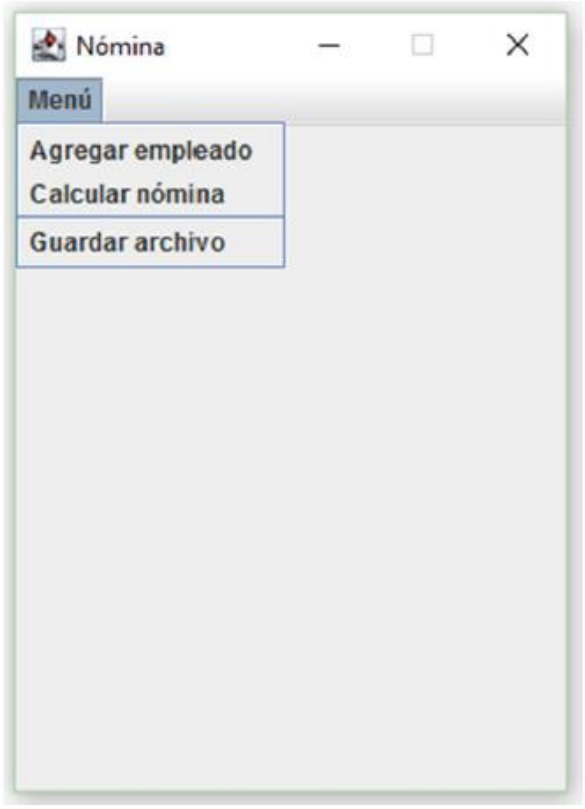
En segundo lugar, la clase VentanaNomina define una ventana que muestra en formato de tabla los principales datos de los empleados y calcula el total de la nómina. Para ello, la clase VentanaNomina cuenta con atributos para definir los diferentes componentes gráficos de la ventana: un contenedor de componentes gráficos (*Container*); etiquetas para identificar los datos a mostrar y calcular (*JLabel*); una tabla para presentar en forma resumida los datos principales de los empleados (*JTable*) y un botón para calcular el total de la nómina (*JButton*). La clase VentanaNomina tiene un constructor y un método para generar la ventana con sus componentes (inicio).

La clase VentanaNomina para generar la tabla de empleados se relaciona con la clase ListaEmpleados por medio de una relación de asociación y a través del atributo lista de empleados, el cual es el nombre de un rol (extremo) de la relación de asociación. La clase ListaEmpleados es una clase contenedora de empleados. Para ello, cuenta con dos atributos: el total de la nómina y la lista de empleados, esta se expresa por medio de una relación de agregación con la clase Empleado. Esta relación de agregación se denota por medio de una línea continua con un rombo blanco en el extremo de la relación que se vincula con la clase ListaEmpleados (que representa el todo). La clase ListaEmpleados cuenta con un constructor y con métodos para agregar un empleado a la lista de empleados para calcular el total de la nómina, obtener la matriz de empleados y convertir a texto la información de la lista de empleados (utilizados para generar el *JTable* en VentanaNomina).

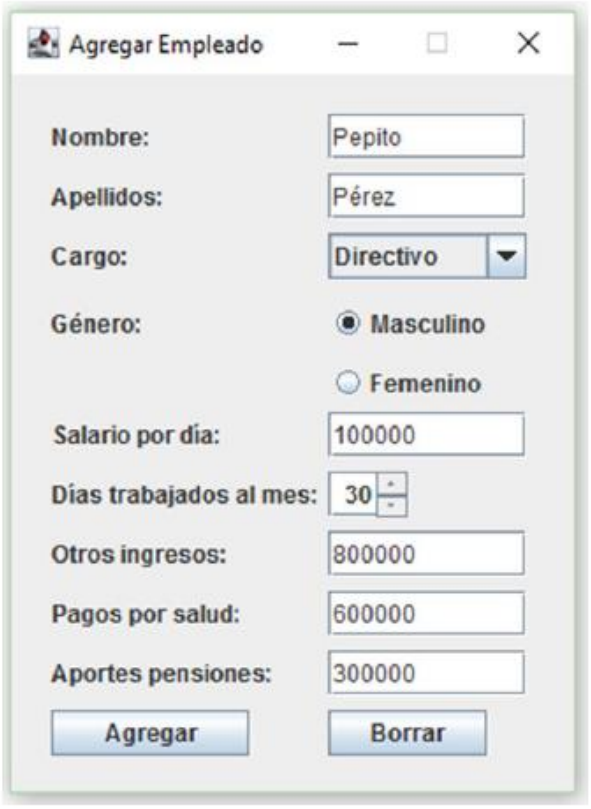
La clase Empleado tiene los siguientes atributos: nombre, apellidos, salario por día, otros ingresos, pagos por salud, aporte pensiones, días trabajados, cargo y género. La clase Empleado cuenta con un constructor que inicializa los atributos del empleado a crear y tiene métodos *get* y *set* para cada atributo. También cuenta con un método para calcular la nómina del empleado.

Diagrama de objetos

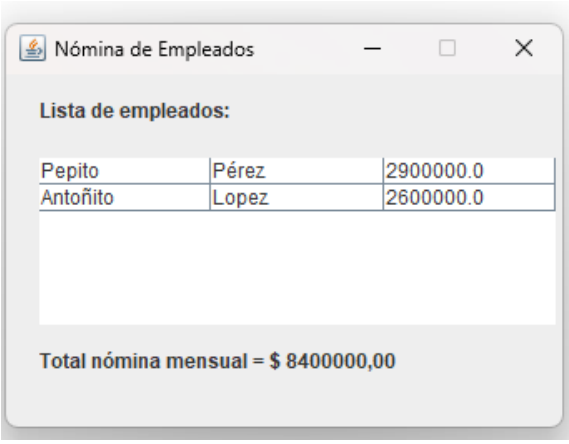
Ejecución del programa



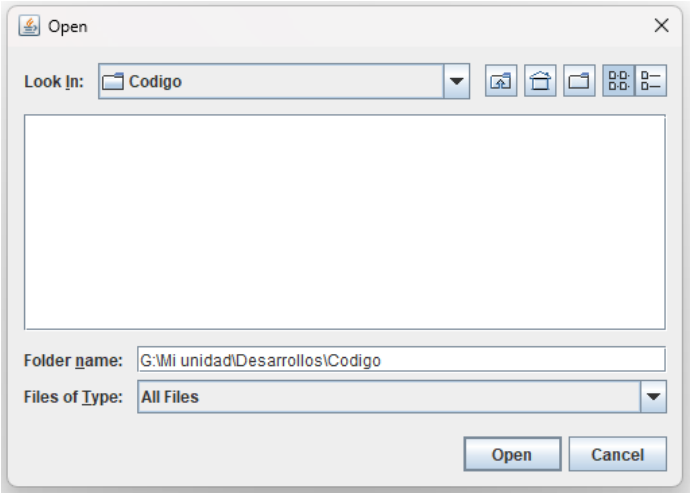
a) Ventana Principal



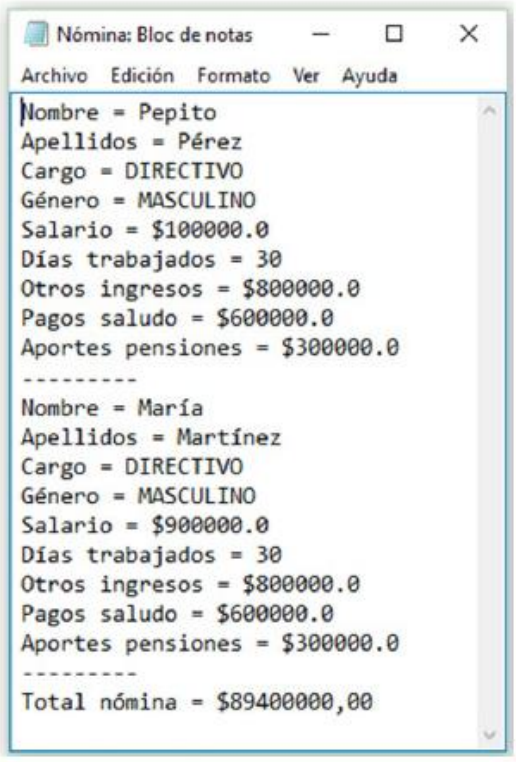
b) Agregar empleado



c) Calcular nómina



d) Guardar Archivo



e) Guardar archivo

Ejercicios propuestos

- Agregar las funcionalidades editar y eliminar empleados al programa.
- Agregar la funcionalidad guardar y leer los datos de la nómina de la empresa, pero como un archivo binario, no de texto.

Instrucciones del ejercicio

Clase	Método	Descripción
JFrame	*JFrame()*	Constructor de la clase *JFrame*.
	void setTitle(String título)	Establece el título de la ventana con el *String* especificado.
	void setSize(int x, int y)	Cambia el tamaño del componente para que tenga una anchura *x* y una altura *y*.
	void setLocationRelativeTo (Component c)	Establece la ubicación de la ventana en relación con el componente especificado.
	void setDefaultCloseOperation (opciones)	Usado para especificar una de las siguientes opciones del botón de cierre: EXIT_ON_CLOSE, HIDE_ON_CLOSE, DISPOSE_ ON_CLOSE o DO_NOTHING_ ON_CLOSE.
	void setResizable(boolean resizable)	Para evitar que se cambie el tamaño de la ventana.
	void setVisible(boolean b)	Muestra u oculta la ventana según el valor del parámetro b.
	void setJMenuBar(JMenuBar menubar)	Establece una barra de menú para la ventana.
ActionListener	*void actionPerformed(actionEvent e)*	Se invoca cuando ocurre un evento.
Container	*Container getContentPane()*	Retorna el objeto *ContentPane* de la ventana.
	void setLayout(LayoutManager mgr)	Establece el *layout* de la ventana.
	Component add(Component comp)	Añade el componente especificado al final del contenedor.
Component	*void addActionListener(this)*	Añade un oyente de eventos al componente actual.
	void setBounds(int x, int y, int ancho, int alto)	Mueve y cambia el tamaño del componente.
JLabel	*JLabel()*	Constructor de la clase *JLabel*.
	void setText(String text)	Define una línea de texto que mostrará este componente.

Clase	Método	Descripción
JTextField	*JTextField()*	Constructor de la clase *JTextField*.
	String getText()	Retorna el texto contenido en el componente de texto.
JButton	*JButton()*	Constructor de la clase *JButton*.
	void setText(String text)	Define una línea de texto que mostrará este componente.
JMenuBar	*JMenuBar()*	Constructor de la clase *JMenuBar*.
	JMenu add(JMenu c)	Añade el menú especificado al final de la barra de menú.
JMenu	*JMenu()*	Constructor de la clase *JMenu*.
	JMenuItem add(JMenuItem menuItem)	Añade un ítem de menú al final del menú.
JMenuItem	*JMenuItem()*	Constructor de la clase *JMenuItem*.
JSeparator	*JSeparator()*	Constructor de la clase *JSeparator*.
ButtonGroup	*ButtonGroup()*	Constructor de la clase *ButtonGroup*.
	void add(AbstractButton b)	Añade un botón al grupo.
JRadioButton	*JRadioButton(String texto, boolean selected)*	Constructor de un botón de radio con el texto especificado y el estado de selección.
JComboBox	*JComboBox()*	Constructor de la clase *JComboBox*.
JSpinner	*void setModel(SpinnerModel modelo)*	Cambia el modelo que representa el valor del *spinner*.
SpinnerNumberModel	*void setMinimum(Comparable mínimo)*	Cambia el mínimo inferior en la secuencia del *spinner*.
	void setMaximum(Comparable máximo)	Cambia el máximo superior en la secuencia del *spinner*.
	void setValue(Object valor)	Establece el valor actual en esta secuencia del *spinner*.
DefaultTableModel	*DefaultTableModel()*	Constructor de la clase *DefaultTableModel*.
JTable	*JTable(Object[][] datosFila, Object[] nombresColumnas)*	Constructor de *JTable* que presenta los valores en un *array* de dos dimensiones (datosFila) con los nombres de las columnas (nombresColomunas).

Clase	Método	Descripción
JFileChooser	*void setFileSelectionMode(int modo)*	Establece el selector de archivo, de directorio o ambos.
	showOpenDialog(Component padre)	Abre un diálogo de selección de archivo.
	File getSelectedFile()	Retorna el archivo seleccionado.
File	*String getName()*	Retorna el nombre del archivo o directorio.
	boolean createNewFile()	Crea un nuevo archivo vacío.
FileWriter	*FileWriter(File archivo)*	Construye un objeto *FileWriter* a partir de un objeto *File*.
BufferedWriter	*BufferedWriter(Writer salida)*	Construye un flujo de caracteres de salida que utiliza un *búfer* de salida.
	void write(Strings)	Escribe un *String* en el archivo.
	void close()	Cierra el flujo.
Event	*Object getSource()*	El objeto sobre el cual el evento inicialmente ha ocurrido.
JOptionPane	*void showMessageDialog (Component componentePadre, Object mensaje)*	Crea un cuadro de diálogo.
String	*String format(String formato, Object … args)*	Retorna un *String* formateado utilizando el formato y argumentos especificados.
Vector	*Object elementAt(int índice)*	Retorna el elemento en la posición especificada.
Double	*String toString()*	Retorna una representación *String* para el objeto *double*.