Nuevos componentes: JFileChooser, JProgressBar, JColorChooser

JFileChooser

Proporciona una interfaz gráfica para que el usuario pueda elegir archivos o directorios desde el sistema de archivos.

Métodos heredados

- `showOpenDialog`: muestra el cuadro de diáolog de selección de archivo en modo "*Abrir archivo*". Devuelve un valor entero que indica la acción tomada por el usuario.

  Opciones:

  - `JFileChooser.APPROVE_OPTION`: el usuario ha seleccionado un archivo y ha hecho clic en "*Abrir*".
  - `JFileChooser.CANCEL_OPTION`: el usuario ha cancelado la acción.
  - `JFileChooser.ERROR_OPTION`: ocurrió un error.

- `showSaveDialog`: muestra el cuadro de diálogo en modo "*Guardar archivo*".

  Opciones:

  - `JFileChooser.APPROVE_OPTION`: el usuario ha seleccionado un archivo y ha hecho clic en "*Abrir*".
  - `JFileChooser.CANCEL_OPTION`: el usuario ha cancelado la acción.
  - `JFileChooser.ERROR_OPTION`: ocurrió un error.

Métodos

- `getSelectedFile()`: devuelve el archivo o directorio seleccionado por el usuario. Hay que usarlo después de que el usuario haya aprobado la selección (usando `showOpenDialog` o `showSaveDialog`).
- `setFileSelectionMode(int mode)`: establece el tipo de archivo que el usuario puede seleccionar. Existen tres opciones principales:

  - `JFileChooser.FILES_ONLY`: permite seleccionar solo archivos.
  - `JFileChooser.DIRECTORIES_ONLY`: permite seleccionar solo directorios.
  - `JFileChooser.FILES_AND_DIRECTORIES`: permite seleccionar tanto archivos como directorios.

- `setCurrentDirectory(File directory)`: establece el directorio inicial que se mostrará cuando se abra el cuadro de diálogo.
- `setDialogTitle(String title)`: establece el título del cuadro de diálogo. Por defecto, el título es "*Abrir*" o "*Guardar*" dependiendo del método que se utilice.
- `setFileFilter(FileFilter filter)`: permite establecer un filtro para mostrar solo ciertos tipos de archivos (texto o imágenes).

- `setMultiSelectionEnabled(boolean enabled)`: habilita o deshabilita la selección múltiple de archivos. Si se habilita, el usuario puede seleccionar más de un archivo al mismo tiempo.

Código fuente:

```
JFileChooser filechooser = new JFileChooser();

File origen = new File("/");
File des = new File("/");

private void guardarDNIActionPerformed(java.awt.event.ActionEvent evt) {
    int res = filechooser.showOpenDialog(null);
    if (res == JFileChooser.APPROVE_OPTION) {
        des = filechooser.getSelectedFile();

        Path or = Paths.get(origen.getAbsolutePath());
        Path dest = Paths.get(des.getAbsolutePath());

        try {
            Files.copy(or, dest, StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException ex) {
            Logger.getLogger(Participantes.class.getName()).log(Level.SEVERE,
null, ex);
        }
    } else {
        System.out.println("Se ha cancelado la subida");
    }
}

private void subirDNIActionPerformed(java.awt.event.ActionEvent evt) {

    int res = filechooser.showOpenDialog(null);
    if (res == JFileChooser.APPROVE_OPTION) {
        origen = filechooser.getSelectedFile();

        // si queremos que el mismo botón lo suba y lo guarde añadimos esto:
        Path or = Paths.get(origen.getAbsolutePath());
        File destinofichero = new File("C:\\Users\\alba_\\Downloads\\dni",
origen.getName());
        Path dest = Paths.get(destinofichero.getAbsolutePath());
        try {
            Files.copy(or, dest, StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException ex) {
            Logger.getLogger(Participantes.class.getName()).log(Level.SEVERE,
null, ex);
        }
    } else {
        System.out.println("Se ha cancelado la subida");
    }
}
```

Usando **setFileFilter**, que permite seleccionar el tipo de archivos que se deben mostrar en el cuadro de diálogo JFileChooser, en este caso .png, .jpg, .jpeg y .pdf:

```
 //usando un filtro para que nos filtre visualmente los archivos que se pueden
seleccionar.
//solo mostrará los que tengan las extensiones indicadas.
filechooser.setFileFilter(new FileFilter() {
    @Override
    public boolean accept(File file) {
        String name = file.getName().toLowerCase();
        return name.endsWith(".png") || name.endsWith(".jpg")
                || name.endsWith(".jpeg") || name.endsWith(".pdf");
    }

    //Lo que aparecen en Files of type
    @Override
    public String getDescription() {
        return "Imágenes y documentos (*.png, *.jpg, *.jpeg, *.pdf)";
    }
});
```

JProgressBar

Se utiliza para mostrar el progreso de una tarea en curso, como una operación de descarga, carga o procesamiento. Puedes usarlos para mostrar visualmente el avance de una tarea que tiene un porcentaje de progreso determinado.

Métodos

- `setIndeterminate(boolean b)`: establece si la barra de progreso debe ser indeterminada, es decir, si no muestra un valor de progreso específico, sino que muestra una animación de progreso indeterminado (como un ciclo infinito).
- `setOrientation(int orientation)`: establece la orientación de la barra de progreso. Puede ser `JProgressBar.HORIZONTAL` o `JProgressBar.VERTICAL`
- `setForeground(Color c)`: establece el color del área que representa el progreso.
- `setBorderPainted(boolean b)`: establece si se debe pintar el borde de la barra de progreso.
- `setBackground(Color c)`: establece el color de fondo de la barra de progreso.
- `setBorderPainted(boolean b)`: establece si se debe pintar el borde de la barra de progreso.

Código fuente:

```
private JLabel labelGrupoRojos, labelGrupoAmarillos, labelGrupoVerdes,
labelGrupoAzules;
private JProgressBar barraRojos, barraAmarillos, barraVerdes, barraAzules;
```

```
// Grupo Rojos (E.I.)
labelGrupoRojos = new JLabel("Grupo Rojos (E.I.)");
labelGrupoRojos.setFont(new Font("Reem Kufi", Font.BOLD, 16));
labelGrupoRojos.setForeground(Color.BLACK);
labelGrupoRojos.setAlignmentX(Component.LEFT_ALIGNMENT);
labelGrupoRojos.setBorder(BorderFactory.createEmptyBorder(10, 0, 5, 0));

barraRojos = new JProgressBar(0, 100);
barraRojos.setStringPainted(true);
barraRojos.setBackground(Color.LIGHT_GRAY);
barraRojos.setForeground(new Color(220, 20, 60)); // Rojo crimson
barraRojos.setPreferredSize(new Dimension(400, 25));
barraRojos.setMaximumSize(new Dimension(Integer.MAX_VALUE, 25));

// Agregamos componentes al panel
panel.add(labelGrupoRojos);
panel.add(barraRojos);
```

JColorChooser

Proporciona una ventana para seleccionar colores de manera interactiva.

Métodos

- `createDialog(Component c, String title, boolean modal, JColorChooser chooserPane, ActionListener okListener, ActionListener cancelListener)`: muestra el cuadro de diálogo para elegir un color.
- `setColor(Color c)`: establece el color inicial que se mostrará en el cuadro de diálogo cuando se llame a `showDialog()`.
- `getColor()`: obtiene el color elegido.

Código fuente

```
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JPanel;

/**
 *
 * @author alba_
 */
public class PruebaJColorChooser {
```

```
    public static void main(String[] args) {
        JFrame frame = new JFrame("Ejempli de JColorChooser");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);

        //Creamos un panel y un botón
        JPanel panel = new JPanel();
        JButton elegir = new JButton("Elige un color");

        /**
         * cuando se pulse el botón aparecerá un JColorChooser, que nos permitirá
modificar el color del panel
         */
        elegir.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JColorChooser colorChooser = new JColorChooser();
                Color c = colorChooser.getColor();

                ActionListener okListener = new ActionListener(){
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        //si se hace clic en Aceptar
                        Color colorSeleccionado = colorChooser.getColor();
                        panel.setBackground(colorSeleccionado);
                    }
                };

                //si se cancela
                ActionListener cancelListener = (ActionEvent e1) ->{
                    System.out.println("Se ha cancelado la elección del color");
                };

                //creamos el cuadro de diálogo
                JDialog dialogo =JColorChooser.createDialog(frame, "Elige un
color", true, colorChooser, okListener, cancelListener);

                //mostramos el cuadro de diálogo
                dialogo.setVisible(true);
            }
        });

        panel.add(elegir);
        frame.add(panel);
        frame.setVisible(true);
    }
}
```