N6 Sudoku
Consideraciones adicionales de diseño

Consejos prácticos para la construcción de la interfaz gráfica de la aplicación

1. El tamaño sugerido de la ventana principal es de 1000 píxeles de ancho y 550 píxeles de alto.
2. Para incluir una imagen en una etiqueta, se utiliza un `ImagenIcon` de la siguiente manera:

```
JLabel lblImagen = new JLabel ( );
ImageIcon icono = new ImageIcon( rutaImagen );
lblImagen.setIcon( icono );
```

   Donde rutaImagen es la ruta donde se encuentra ubicada la imagen. Por ejemplo, la imagen del banner se encuentra en "./data/imagenes/banner.png".

3. Las imágenes de los iconos de navegación se encuentran en la siguiente ruta: `./data/imagenes/`

   - abajo.png: Imagen del botón para navegar hacía abajo.
   - arriba.png: Imagen del botón para navegar hacía arriba.
   - derAb.png: Imagen del botón para navegar hacía abajo a la derecha.
   - derArr.png: Imagen del botón para navegar hacía arriba a la derecha.
   - derecha.png: Imagen del botón para navegar hacía la derecha.
   - IzqAb.png: Imagen del botón para navegar hacía abajo a la izquierda.
   - izqArr.png: Imagen del botón para navegar hacía arriba a la izquierda.
   - izquierda.png: Imagen del botón para navegar hacía la izquierda.

4. Para centrar una imagen en una etiqueta, utilice setHorizontalAlignment de la siguiente manera:

```
lblImagen.setHorizontalAlignment( JLabel.CENTER );
```

5. Para definir las dimensiones que tiene una imagen usted puede utilizar la siguiente instrucción:

```
ImageIcon icono = new ImageIcon( new ImageIcon(
ruta ).getImage().getScaledInstance( 200 , 200 , Image.SCALE_DEFAULT ) );
```

En esta instrucción se está definiendo el alto y el ancho de la imagen como 200 y 200.

6. Para modificar la apariencia de una etiqueta, existen distintas instrucciones:

   ✓ Para modificar la fuente:

```
label = new JLabel( );
label.setFont( new Font("Helvetica", Font.BOLD, 14) );
```

En el ejemplo se está asignando la fuente Helvética en negrita, de tamaño 14. Casi todas las etiquetas tienen la fuente de java por defecto, excepto el nombre del edificio.

- ✓ Para cambiar la alineación del texto,

```
label.setHorizontalAlignment( JLabel.CENTER );
```

7. Para modificar el color de un campo de texto, debe usar el método setBackground de la siguiente manera:

```
txt.setBackground( Color.CYAN );
```

Este método recibe el color de fondo del campo de texto que se desea colocar. En el caso del ejercicio, para los colores usados existen las consantes ***CYAN***, ***ORANGE***, ***YELLOW*** y ***GREEN*** de la clase Color.

8. Para que haya un espacio entre los elementos en un GridLayout, puede agregarle 2 parámetros adicionales al constructor del distribuidor gráfico, para indicar los espacios, en pixeles, entre cada una de sus zonas.

```
setLayout( new GridLayout( 2, 1, 2, 2 ) );
```

En este caso, se tiene una malla de 2 filas y una columna, y se deja un espacio de 2 píxeles entre las filas y un espacio de 2 filas entre las columnas.

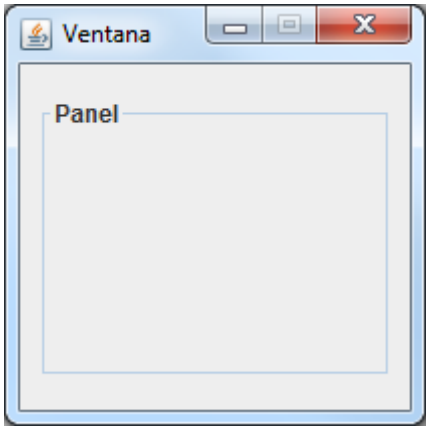
9. Para quitar todos los elementos de un panel puede usar el método `removeAll()`
10. Después de eliminar todos los elementos de un panel y agregar nuevos componentes debe utilizar los métodos `revalidate()` y `repaint()`

```
...

// Actualizar el panel
...
revalidate();
repaint();

...
```

Estos métodos le pueden resultar necesarios al momento de cargar un nuevo tablero con dimensiones diferentes al actualmente cargado.

11. Gráficamente, el tablero de juego se puede implementar con una matriz de `JTextField`. Cada una de las zonas del tablero se puede representar usando un panel auxiliar. Para que cada panel tenga un borde con relieve, se puede usar el tipo de borde `BevelBorder` junto con la constante `BevelBorder.RAISED`. Su uso se ilustra a continuación.

Suponga que tiene la siguiente interfaz gráfica en la cual se incluye un único panel que tiene un borde tipo `TitledBorder`.

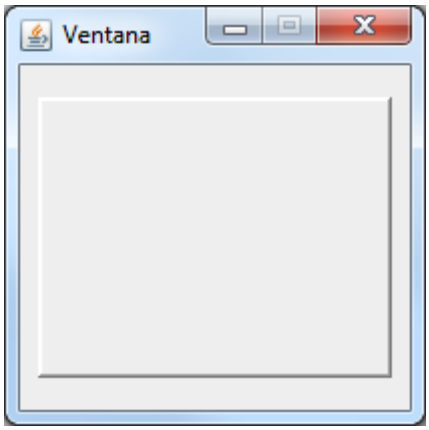


Ventana con un panel cuyo borde es de tipo `TitledBorder`

El fragmento de código que implementaría dicho panel se muestra a continuación:

```
...

// Creación del panel central

JPanel panel = new JPanel();
Border b = new TitledBorder("Panel");
panel.setBorder(b);
add(panel, BorderLayout.CENTER);

...
```

Ahora, desea cambiar el borde del panel para que no tenga título y tenga la siguiente apariencia (borde con relieve):



Ventana con un panel cuyo borde es de tipo `BevelBorder`

El fragmento de código que implementaría el panel con la nueva apariencia se muestra a continuación. Note que lo único que cambia es el tipo de borde usado (ahora se usa `BevelBorder`).

```
 ...

 // Creación del panel central

 JPanel panel = new JPanel();
 Border b = new BevelBorder( BevelBorder.RAISED );
 panel.setBorder(b);
 add(panel, BorderLayout.CENTER);

...
```

12. Opcionalmente, para hacer que el sudoku se vea siempre cuadrado y centrado sin importar sus dimensiones puede agregar un borde al panel que contiene las casillas de la siguiente manera:

```
 ...

TitledBorder borde = new TitledBorder( "Tablero" );
int width = panelCasillas.getWidth();
int height = panelCasillas.getHeight();
int max = Math.max(width, height);
max = sudoku.darTamanioTablero()*45;
max= Math.min(max, width);
max=Math.min(max, height);
int hMargin = Math.abs((max - height)/2);
int wMargin = Math.abs((max-width)/2);
panelCasillas.setBorder(new CompoundBorder(borde,
BorderFactory.createEmptyBorder(hMargin, wMargin, hMargin,
wMargin)));

...
```

Recomendaciones

- Recuerde utilizar los mensajes de las excepciones para mostrar la información respecto al error ocurrido. No vuelva a escribir el mensaje.
- Tenga en cuenta que el ejercicio debe funcionar sin problemas si se cambia el archivo desde donde se cargan los sudokus.

- Se recomienda tener en la clase principal de la interfaz un método que actualice la información del juego, que debe ser llamado cada vez que se realice un movimiento.