Documento de Sustentación del Código: `SplashScreen.java`

## Descripción General

El código presentado corresponde a una ventana de \*\*Splash Screen\*\* que se muestra al iniciar una aplicación. Esta pantalla contiene una barra de progreso que indica el porcentaje de carga de los módulos de la aplicación antes de la transición hacia el formulario de inicio de sesión (`FrmLogin`). Se utilizan diferentes componentes gráficos como imágenes de fondo, etiquetas, y una barra de progreso.

## Objetivo del Código

El \*\*Splash Screen\*\* se utiliza para proporcionar una experiencia visual atractiva mientras los módulos de la aplicación se cargan. El objetivo del código es mostrar una ventana sin bordes que carga diversos elementos de la aplicación y luego redirige al formulario de inicio de sesión una vez que la carga ha terminado.

## Componentes Utilizados

1. \*\*JFrame\*\*: La clase principal hereda de `javax.swing.JFrame` para definir una ventana gráfica.

2. \*\*JProgressBar\*\*: Se utiliza para mostrar el progreso de la carga de la aplicación.

3. \*\*JLabel\*\*: Para mostrar texto e imágenes.

4. \*\*ImageIcon\*\*: Se utiliza para cargar y escalar imágenes que se establecen como fondo y otros elementos visuales.

5. \*\*Thread.sleep()\*\*: Simula el proceso de carga mediante la pausa temporal del hilo principal.

## Análisis del Código

### 1. Constructor de la Clase `SplashScreen`

El constructor de la clase `SplashScreen` contiene la inicialización de los componentes y algunas configuraciones clave:

- \*\*Sin bordes\*\*: `setUndecorated(true)` elimina los bordes de la ventana.

- \*\*Tamaño y visibilidad\*\*: La ventana tiene un tamaño fijo de 900x500 píxeles, es visible y no redimensionable.

- \*\*Esquinas redondeadas\*\*: `setShape(new RoundRectangle2D.Double(...))` permite redondear las esquinas de la ventana con un radio de 20.

- \*\*Imagen de fondo\*\*: Se carga una imagen desde el directorio `src/images/bg-gradiente.jpg` y se ajusta al tamaño del `jLabel\_Wallpaper` para ser utilizada como fondo de la ventana.

### 2. Inicialización de Componentes Gráficos (`initComponents`)

Este método se encarga de crear y posicionar los diferentes elementos de la interfaz gráfica:

- \*\*Imagen de fondo\*\*: `BackgroungImage1` contiene una imagen (`/images/robot-assistant.png`) que se coloca en la parte derecha de la ventana.

- \*\*Barra de progreso\*\*: `LoadingBar` se posiciona y actualiza su valor según el progreso simulado.

- \*\*Etiquetas\*\*: `LoadingLabel` muestra mensajes de estado sobre el progreso, y `LoadingValue` muestra el porcentaje actual de la carga.

### 3. Método `main`

El método `main` contiene la lógica que simula el progreso de carga:

- \*\*Bucle de carga\*\*: Se utiliza un bucle `for` para simular la carga de la aplicación, actualizando el porcentaje en la barra de progreso y mostrando mensajes informativos en `LoadingLabel`.

Los mensajes cambian en función del valor de la variable `i`:

- \*\*10%\*\*: "Turning On Modules..."

- \*\*20%\*\*: "Loading Modules..."

- \*\*50%\*\*: "Connecting to Database..."

- \*\*70%\*\*: "Connecting Successful..."

- \*\*80%\*\*: "Launching Application..."

- \*\*Transición al login\*\*: Cuando la barra llega al 100%, se oculta el `SplashScreen` y se muestra el formulario de inicio de sesión `FrmLogin`.

### 4. Manejo de Excepciones

Se implementa un bloque `try-catch` para controlar posibles excepciones durante el proceso de carga. En caso de error, se muestra un mensaje emergente con la información de la excepción usando `JOptionPane.showMessageDialog()`.

## Aspectos Destacados

1. \*\*Optimización de imágenes\*\*: La imagen de fondo se ajusta dinámicamente al tamaño del componente `jLabel\_Wallpaper` utilizando el método `getScaledInstance()`, lo que garantiza que la imagen siempre se vea correctamente.

2. \*\*Ventana sin bordes\*\*: El uso de `setUndecorated(true)` y el redondeo de las esquinas con `RoundRectangle2D.Double` mejora la estética de la ventana, dándole un diseño más moderno y minimalista.

3. \*\*Experiencia de usuario\*\*: A través de la barra de progreso y los mensajes de carga, el usuario tiene una indicación clara del estado actual de la aplicación, lo que mejora la percepción de la carga.

## Posibles Mejoras

1. \*\*Cargar imágenes de forma dinámica\*\*: Utilizar rutas relativas o permitir la configuración de las imágenes desde fuera del código para mayor flexibilidad.

2. \*\*Temporización dinámica\*\*: En lugar de usar `Thread.sleep()` para simular la carga, se puede implementar un sistema real de seguimiento del progreso de los módulos cargados, especialmente si existen procesos de carga reales como conexión a bases de datos o inicialización de servicios.

3. \*\*Animaciones adicionales\*\*: Se podrían agregar transiciones suaves para la carga de elementos, como una animación al aparecer o desaparecer la ventana del Splash Screen.

4. \*\*Internacionalización\*\*: Incluir soporte para diferentes idiomas en los mensajes de carga a través de un archivo de propiedades (`.properties`).

## Conclusión

Este código proporciona una ventana de inicio con un diseño estético y funcional, mejorando la experiencia de usuario mediante la visualización de una barra de progreso y mensajes dinámicos. A través del `SplashScreen`, se prepara al usuario para el inicio de la aplicación mientras los módulos se cargan en segundo plano.