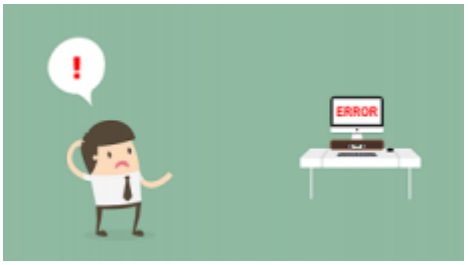
Usabilidad y accesibilidad

Los conceptos de usabilidad o experiencia de usuario y accesibilidad tienen una importante relación con las interfaces de usuario. El estándar internacional ISO/IEC 25000 sobre los requisitos y la evaluación de calidad del software indica que la usabilidad *se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.*



La accesibilidad se refiere a que las aplicaciones software, las herramientas y las tecnologías están diseñadas y desarrolladas para que las personas con discapacidad puedan usarlas. Algunas características de los usuarios ante las que deben estar adaptadas las interfaces gráficas pueden ser:

- Discapacidades visuales: desde baja visión hasta ausencia completa.
- Discapacidades de movimiento: dificultad para mover un ratón o pulsar varias teclas al mismo tiempo.
- Deficiencias auditivas: desde ser capaz de oír algunos sonidos pero no distinguir las palabras habladas, hasta ausencia de audición.

- Trastornos Cognitivos y del Lenguaje: Es difícil la exposición compleja o inconsistente, o la selección pobre de palabras usando ordenadores para estos usuarios.
- Nuestras interfaces gráficas deben ser diseñadas atendiendo a cualquier tipo de usuario.

Principios de usabilidad y accesibilidad

Aunque los dos conceptos no son idénticos, tienen ciertos aspectos comunes. A continuación se enumeran algunos:

- **Identificación de controles:** Un usuario necesita identificar para qué sirve cada control de la interfaz y qué información hay que introducir o seleccionar en cada caso. Debemos acompañar etiquetas o mensajes desplegables para identificar de forma visual la finalidad del control o la información que el usuario debe introducir.Esta identificación se debe hacer mediante texto antes que mediante imágenes, aunque no impide el uso de imágenes para complementar la identificación.
- **Distribución de controles:** Una interfaz gráfica debe ofrecer una estructura o distribución de los controles que resulte comprensible para el usuario. Asociar o agrupar los controles atendiendo a su finalidad, u ordenarlos dependiendo de su importancia o frecuencia de uso, son algunos ejemplos.

- **Sencillez de la interfaz:** Las interfaces gráficas más modernas persiguen la sencillez debido a varias razones: pantallas más pequeñas, mejora de la usabilidad y experiencia de usuario y división de subinterfaces por funcionalidad. Para ello se utilizan pestañas que nos permiten cambiar entre vistas de información o botones que muestran una nueva ventana de la aplicación. El

objetivo es no saturar al usuario con multitud de controles dentro de una misma sección.
- **Acceso a controles frecuentes:** las opciones de uso más habitual no deben tener un acceso rápido dentro de nuestra interfaz gráfica, y que no se alberguen con mucha profundidad dentro de menús o secciones.
- **Uso del teclado:** facilitar el acceso eficiente del teclado a todas las funciones de la aplicación. Algunos usuarios pueden ser incapaces de usar un ratón, y muchos «usuarios avanzados» prefieren usar el teclado de todos modos.
- **Opciones para deshacer:** permitir deshacer acciones, o mantener un historial de acciones realizadas para poder corregir o volver al estado original ante posibles acciones no intencionadas.
- **Asistencia desde la interfaz:** mostrar mensajes de ayuda que indiquen al usuario qué puede hacer, o cómo debe usar ciertos controles. Impedir el uso incorrecto de la aplicación ante posibles errores.
- **etc…**

En la guía de accesibilidad para desarrolladores de GNOME se indican muchas orientaciones sobre estos aspectos.

Aspecto Look&Feel

El look and feel de una interfaz gráfica en Java se encarga del aspecto de los componentes gráficos. Como Java es un lenguaje multiplataforma por defecto utiliza una estética independiente del sistema en el que se ejecute la aplicación.

Para modificar esta estética utilizamos la clase **UiManager** junto con su método setLookAndFeel(), que recibe un `String` en el que se indica el ***lookandfeel*** que se quiere emplear.En caso de modificar el *look and feel* de una aplicación, s**e debe establecer al inicio del método main() que inicie la aplicación.**

Diferentes aspectos look & feel

Look&Feel dependiente o independiente del sistema

La estética *look and feel* por defecto se conoce como *cross-platform look and feel* también llamada *metal look and feel*. Pero en algunos casos quizás nos interese utilizar el *look and feel* del sistema en el que se ejecuta la aplicación.

```
//Usar cross-platform L&F, ó metal L&F
UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());

//Usar el look and feel del sistema anfitrión
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName();
```

*El método setLookAndFeel() lanza excepciones que debemos controlar con un try-catch.

Otros Look&Feel

También podemos modificar el look and feel mediante el pasando como parámetro el String que lo identifica. Los distintos aspectos que podemos usar son:

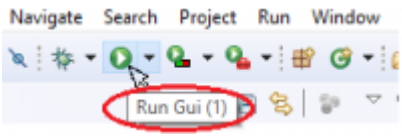
Metal o Cross-Platform: Clase javax.swing.plaf.metal.MetalLookAndFeel
Windows: Class com.sun.java.swing.plaf.windows.WindowsLookAndFeel
Windows Classic: Class com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel
Nimbus: Clase javax.swing.plaf.nimbus.NimbusLookAndFeel
CDE/Motif: Clase com.sun.java.swing.plaf.motif.MotifLookAndFeel

```
try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

//UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

//UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

//UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");

//UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsClasicLookAndFeel");

} catch (ClassNotFoundException e) {
   e.printStackTrace();
} catch (InstantiationException e) {
   e.printStackTrace();
} catch (IllegalAccessException e) {
   e.printStackTrace();
} catch (UnsupportedLookAndFeelException e) {
   e.printStackTrace();
}
```

Iconos intuitivos

Los botones de la mayoría de aplicaciones, no son botones que contengan texto, sino que contienen iconos. Del mismo modo los elementos de los menús, aunque sí que contienen texto, también indican su función mediante un icono .Debemos seleccionar las imágenes que contienen dichos botones de forma que representen las funciones que realizan.

Etiquetas ToolTips de ayuda



Un tooltip es una ayuda visual que nos permite insertar un texto a modo de explicación o ayuda sobre un componente o control de la aplicación. Este texto aparecerá automáticamente, al pasar el ratón por encima del componente.

```
JButton btnMensajeError = new JButton("Abrir fichero");


//Incluimos una ayuda tooltip sobre el botón

btnMensajeerror.setToolTipText("Opción de abrir fichero");
```