ANEXOS

1. Instalación del Framework de OSCAR.

Como hemos comentado anteriormente el framework de OSCAR es la implementación que hemos elegido del framework de OSGi para nuestro proyecto.

Veremos en este apartado los pasos necesarios para la correcta instalación del framework.

Podemos acceder al núcleo del framework de OSCAR en la siguiente dirección WEB: http://oscar.objectweb.org

Una vez hayamos entrado en dicha página debemos acceder a la zona de download y bajar la última versión disponible.

Es necesario tener instalado en nuestro sistema una versión reciente de JDK o del JRE de Sun Microsystems.

Si sólo queremos visualizar aplicaciones y no desarrollarlas nos bastará con tener instalada una versión reciente de JRE.

Una vez descargado el paquete con la versión más reciente de OSCAR debemos realizar su instalación de la siguiente manera:

java -jar oscar-1.0.0.jar

Si la versión que hemos obtenido es distinta a la 1.0.0 simplemente debemos sustituir dicha numeración por la de la versión OSCAR que hayamos obtenido.

La ejecución del comando anterior nos redirige a una ventana como la siguiente:

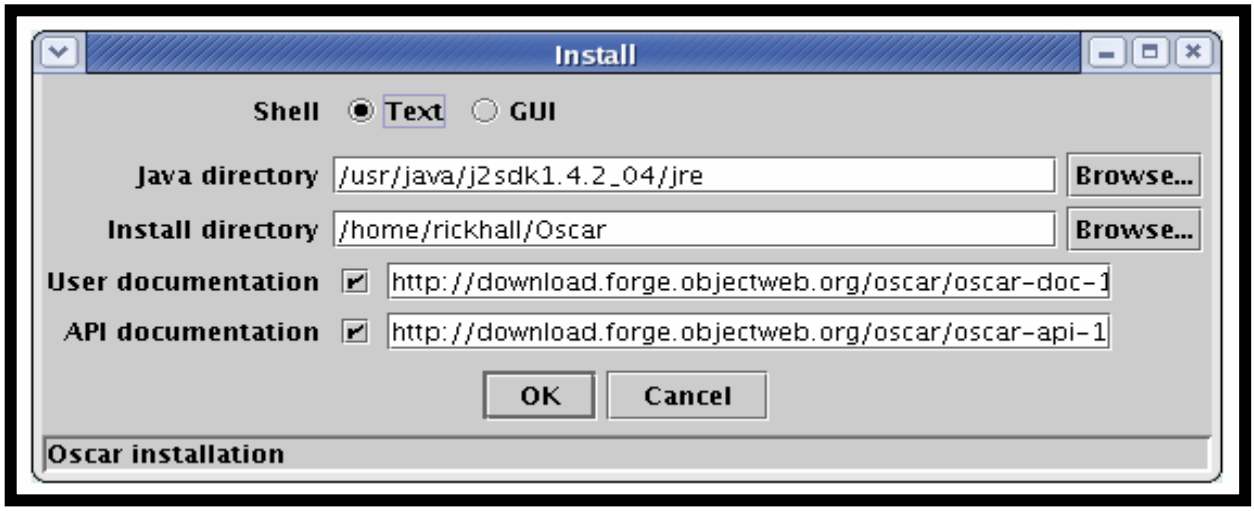

Figura 18: Menú de instalación del framework de OSCAR.

En este menú podemos elegir el tipo de instalación que queremos usar. Podemos elegir un Shell tipo GUI que nos facilitará la comprensión de los comandos del interfaz.

En el Java directory debemos especificar la ruta donde tenemos instalado nuestro jre.

En Install directory especificamos la situación del directorio elegido para la instalación del framework.

Por último si tenemos marcados los campos User documentation y API documentation el programa instalador obtendrá información referente a las librerías e información de usuario del framework de OSCAR.

Una vez finalizada la instalación debemos comprobar en el directorio elegido si existe la siguiente lista de archivos:

- LICENSE.txt – Licencia de OSCAR.
- oscar.bat – Lanza el shell de OSCAR sobre Window.
- oscar.sh – Lanza el shell de OSCAR en linux.
- example.policy – Ejemplo.
- src.jar – Fuentes de todas las librerías OSCAR.
- lib/ - Librerías necesarias para ejecutar OSCAR.
- bundle/ - Bundles necesarios para ejecutar el Shell de OSCAR.
- doc/ - Documentación de usuario y del API si lo elegimos en la instalación.

2.- Elección de JDOM como parseador de XML

Una de las decisiones que más se ha valorado a la hora de realizar el tratamiento de los ficheros XUL es la elección del parseador.

En la actualidad existen una gran cantidad de soluciones para el parseo de documentos XML, aunque todas ellas pueden dividirse en dos grandes grupos.

- **Parseadores tipo DOM:** Los parseadores tipo DOM recorren el fichero XML y van generando en memoria un árbol de nodos con todos los elementos que contiene el fichero XML. Esto permite una vez generado el árbol acceder a cada uno de los nodos y realizar modificaciones. Su principal desventaja es que para archivos XML grandes el tamaño del árbol en memoria es muy elevado.

- **Parseadores tipo SAX:** Los parseadores tipo SAX están basados en eventos. Es decir, van buscando determinados elementos como inicios de etiquetas, finales de etiquetas, para lanzar eventos que deben ser manejados. Esto hace que los parseadores tipo SAX sean mucho mas sencillos y menos pesados que los parseadores tipo DOM, ya que no requieren generar un árbol en memoria. Su principal desventaja es que no permiten la modificación del fichero XML una vez que éste ha sido recorrido.

- **Parser JDOM:** Ninguno de los parseadores anteriores han sido diseñados para tener en cuenta las peculiaridades del lenguaje de programación Java.
  JDOM es una capa por encima de un parseador tipo DOM o SAX que nos permite abstraernos del tipo de parser usado para el tratamiento del los ficheros XML. Debemos dejar claro que JDOM no es un parseador, si no que necesita de uno de ellos para poder tratar archivos XML.

Podemos ver en la siguiente estructura donde situamos JDOM:

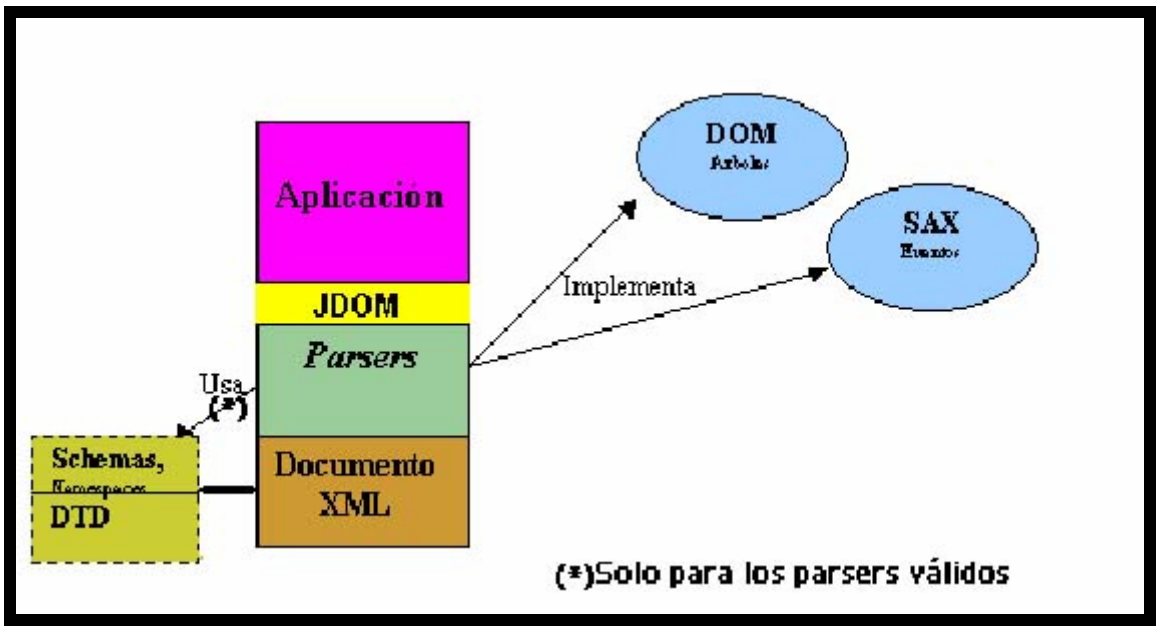

Figura 19: Capa JDOM como capa de abstracción de los diferentes parseadores.

Una vez analizadas las distintas opciones parece lógica la elección de JDOM como nuestro capara sobre la que desarrollar nuestras clases de tratamiento de XML. Así podremos cambiar de parseador sin necesidad de reconstruir la aplicación. Como parseador de nuestra aplicación hemos usado el parseador Xerces (SAX) integrado en el paquete JDOM.

Se ha optado por usar un parser SAX, ya que con JDOM tenemos la misma interfaz que con un parser DOM y el consumo de memoria es menor. Este requisito de la memoria nos es importante en un pc pero si puede ser crítico en una PDA o dispositivo similar.

3. El paquete Charva

Otra de las decisiones que se ha tenido que tomar en el desarrollo de nuestro proyecto ha sido la elección de la librería usada para la creación de los widgets de la vista texto.

Dos han sido las opciones evaluadas:

- **Paquete JCURSES:** JCurses es un API para la generación de aplicaciones Java sobre terminales texto. El paquete JCurses ha sido desarrollado sobre las "curses" del sistema Linux.

- **Paquete Charva:** El paquete Charva es un API para el desarrollo de aplicaciones Java sobre terminales tipo texto. Su principal ventaja sobre el paquete anterior es que está escrito siguiendo la estructura del paquete Swing y AWT, por lo que una vez escrita una interfaz en Swing o AWT es muy fácil generar la misma interfaz en modo texto. Esto facilita enormemente la tarea de generar la vista texto una vez tenemos la vista swing.

Podemos ver aquí dos imágenes de sendas interfaces escritas con Swing y con Charva.

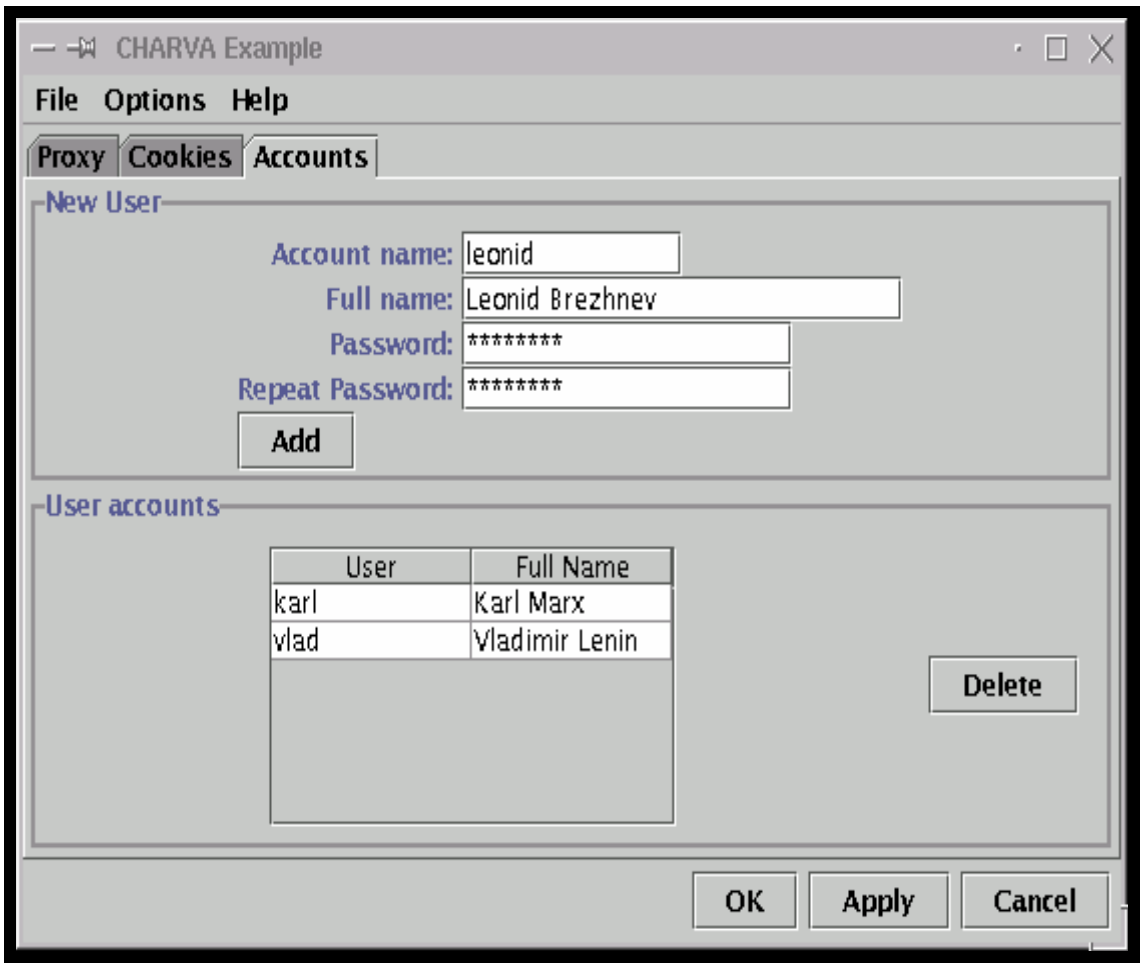

Figura 20. Podemos ver aquí el interfaz de usuario de una aplicación escrito usando el paquete Swing.

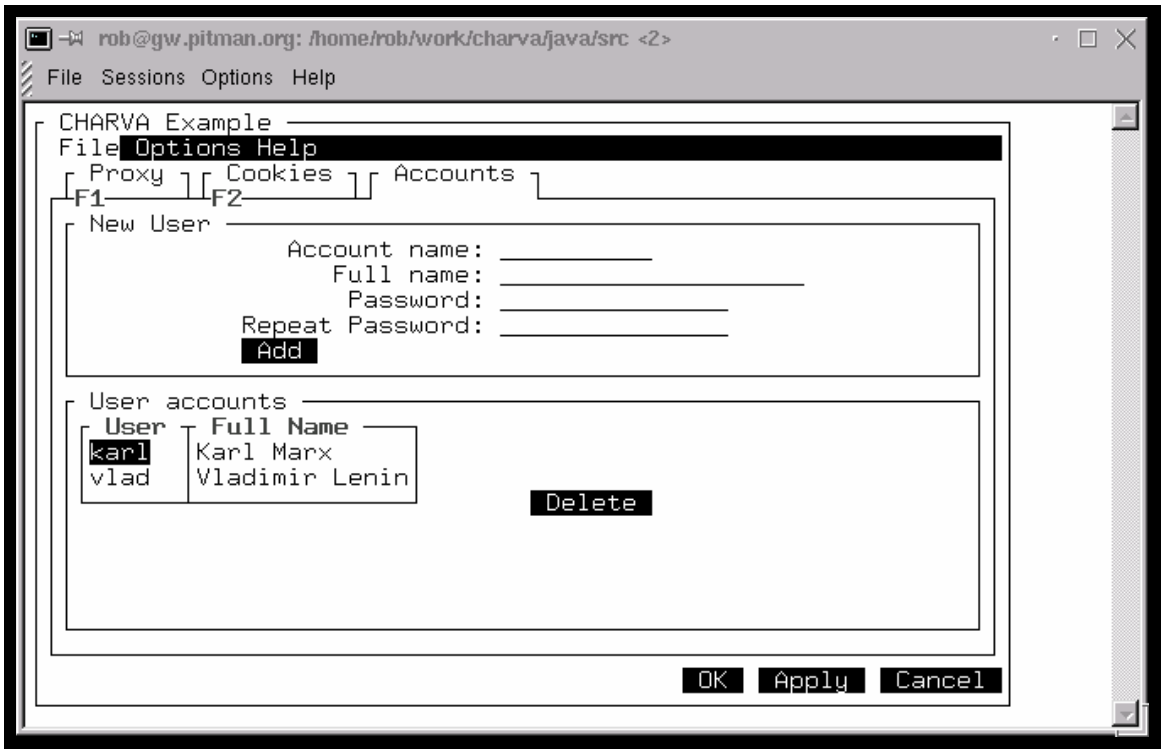

Figura 21. Podemos ver la misma aplicación usando el paquete charva en vez del paquete Swing.

4. ¿Cómo añadir un nuevo Wigdet a una vista existente?

En este apartado vamos a explicar el procedimiento para añadir un nuevo widget a una de las vistas del interfaz existente.

Tomaremos como ejemplo la vista tipo swing y veremos el procedimiento para añadir el widget Button al interfaz.

1. Añadir a la clase *IntergerID* del paquete utils una nueva entrada para el nuevo Widget.

```
13
14 public class IntegerID {
15
16
17    private static final int window = 0;
18    private static final int button = 1;
19    private static final int label  = 2;
20    private static final int hbox   = 3;
```

```
44
45          if(id == "button")
46             return_value = button;
47
48          if(id == "label")
49             return_value = label;
50
51          if(id == "hbox")
52             return_value = hbox;
53
```

2. Añadir en la estructura “Switch(etiqueta)” de la clase *SwingEngine* una nueva entrada para nuestro nuevo widget.

```
public void engine(Element element) throws MalformedURLException {

   name = element.getName();// Obtenemos el tipo de elemento

   switch(IntegerID.returnID(name)){


     // Para cada uno de los elementos simples sólo tenemos que hacer una llamada
     // a su clase e introducir la referencia del objeto en la Hashtable junto a su id.
     case window : {

       Parser.set_of_objects.put(element.getAttributeValue("id"),new WindowBox(element));
       break;
     }
     case button :{

       Parser.set_of_objects.put(element.getAttributeValue("id"),new Button(element));
       break;
     }
```

Como podemos ver en la figura anterior es en esta zona del programa donde se realiza el registro de la referencia al widget en una tabla junto a su identificador.
Si estuviésemos en el caso de un elemento tipo "padre" tendríamos además que realizar la llamada a los siguientes métodos:

```
case hbox : {

     Hbox element_box = new Hbox(element);

    Parser.set_of_objects.put(element.getAttributeValue("id"), element_box);

     element_box.addMyChildren(element);
     element_box.addToMyParent(element);
     element_box.repainMyParent(element);

     break;
}
```

Con la ejecución de los tres métodos marcados en rojo, añadimos al widget actual todos aquellos widget que cuenten de él, a la vez que hacemos colgar la estructura de su elemento padre.

3. Implementación del interfaz widget. La implementación de este interfaz suele hacerse por herencia de la clase WidgetParent o WidgetChild dependiendo de si el elemento es un elemento "padre" o un elemento "hijo" respectivamente. En el caso que nos ocupa, Widget Button, crearemos la clase Button en el paquete widgets y heredaremos la clase WidetChild.

```
public class Button extends WidgetChild{

  private JButton button = new JButton();
  private Element element;
```

4. Implementación de los métodos del interfaz Widget. Dentro de la clase Button debemos implementar todos aquellos métodos del interfaz Widget que no hayan sido implementados en la clase WidgetChild, o que deban ser sobrescritos.

5. Implementación de los métodos manejadores de las etiquetas definidas en la descripción XUL del widget Button. Debe existir un método que maneje cada una de las etiquetas existentes en la descripción XUL del elemento, y que realice la modificación necesaria al widget antes de dibujarse sobre la interfaz.

6. Implementar los manejadores de eventos de la clase encapsulada por el Widget Button (JButton), para que redirijan el tratamiento del evento a los manejadores especificados en la descripción XUL del interfaz.

```
public void setMouseClickedHandlerName(){
  Mouse_Clicked_Handler = element.getAttributeValue("action");
  }


//! El método MouseClickedHandler() es el manejador local del evento de click.
/*! Este método es el encargado de lanzar el manejador de evento seleccionado por el
    método setMouseClickedHandlerName().
*/
 void MouseClickedHandler(){
   EventThrowing thrower = new EventThrowing();
   thrower.setEventContainer("interfaz.EventHandlers");
   thrower.setEvent(Mouse_Clicked_Handler);
   thrower.throwing();
 }
```

7. Añadir los manejadores de eventos en la clase EventHandlers del paquete interfaz.

Cumplidos los siete pasos definidos anteriormente, el elemento queda completamente operativo para poder ser usado por el servicio interfaz de usuario para su representación como parte de la UI de cualquier aplicación.

5. Diagrama de clases.

Lista de paquetes del proyecto OSGi-UI

charva.awt
charva.awt.event
charva.awt.util
charvax.swing
charvax.swing.border
charvax.swing.event
charvax.swing.table
GenericElements
interfaz
java.awt
java.awt.event
java.io
java.lang.reflect
java.net
java.util
javax.swing
org.jdom
org.jdom.input
org.jdom.output
utils
widgets
widgetsTexto

Diagrama de clases del proyecto OSGi-UI

- widgets.ButtonMouseAdapter
- interfaz.Error
- interfaz.EventHandlers
- interfaz.EventThrowing
- GenericElements.GenericEngine
    - interfaz.SwingEngine
    - interfaz.TextEngine
- GenericElements.GenericParser
- utils.ImagePanel
- utils.IntegerID
- utils.JDomUtils
- interfaz.Parser
- interfaz.ParserTexto
- interfaz.Reference
- interfaz.SwingTextTutorial
- GenericElements.widget
    - GenericElements.WidgetChild
        - widgets.BoxFiller
        - widgets.Button
        - widgets.Images
        - widgets.Label
        - widgets.ListItem
        - widgets.MenuItem
        - widgets.RadioButton
        - widgets.TextBox
    - GenericElements.WidgetChildTexto
        - widgetsTexto.Button
        - widgetsTexto.Label
        - widgetsTexto.ListItem
        - widgetsTexto.MenuItem
        - widgetsTexto.RadioButton
        - widgetsTexto.TextBox
    - GenericElements.WidgetParent
        - widgets.Hbox
        - widgets.Listbox
        - widgets.MenuList
        - widgets.MenuPopup
        - widgets.RadioGroup
        - widgets.Vbox
        - widgets.WindowBox
    - GenericElements.WidgetParentTexto
        - widgetsTexto.Hbox
        - widgetsTexto.Listbox
        - widgetsTexto.MenuList
        - widgetsTexto.MenuPopup
        - widgetsTexto.RadioGroup
        - widgetsTexto.Vbox
        - widgetsTexto.WindowBox

Descripción de las clases del proyecto OSGi-UI

widgets.BoxFiller	*Clase **BoxFiller** implementa la interfaz Widget*
widgets.Button	*Clase **Button** implementa la interfaz Widget*
widgetsTexto.Button	*Clase **Button** implementa la interfaz Widget*
widgets.ButtonMouseAdapter	
interfaz.Error	*Esta clase gestiona todos los errores ocurridos durante la ejecución de la interfaz*
interfaz.EventHandlers	*Esta clase contiene los manejadores de todos los eventos que puedan producirse en la interfaz*
interfaz.EventThrowing	
GenericElements.GenericEngine	***GenericEngine** representa la interfaz que debe implementar cualquier clase Engine*
GenericElements.GenericParser	***GenericParser** representa la interfaz que debe implementar cualquier parseador de cualquiera de la vistas*
widgetsTexto.Hbox	*Clase **Hbox** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgets.Hbox	*Clase **Hbox** implementa la interfaz widget y hereda los métodos de WidgetParent*
utils.ImagePanel	***ImagePanel** es una extension de JPanel para poder mostrar una foto de fondo*
widgets.Images	*Clase **Images** implementa la interfaz Widget*
utils.IntegerID	*Clase **IntegerID** devuelve un identificador entero dependiendo del identificador String que se le pasa*
utils.JDomUtils	*Clase **JDomUtils** recoge todas aquellas facilidades interesantes a la hora de trabajar con con ficheros XML*
widgetsTexto.Label	*Clase **Label** implementa la interfaz Widget*
widgets.Label	*Clase **Label** implementa la interfaz Widget*
widgetsTexto.Listbox	*Clase **Listbox** implementa la interfaz Widget*
widgets.Listbox	*Clase **Listbox** implementa la interfaz Widget*
widgetsTexto.ListItem	*Clase **ListItem** implementa la interfaz Widget*
widgets.ListItem	*Clase **ListItem** implementa la interfaz Widget*
widgetsTexto.MenuItem	*Clase **MenuItem** implementa la interfaz Widget*
widgets.MenuItem	*Clase **MenuItem** implementa la interfaz Widget*
widgetsTexto.MenuList	*Clase **MenuList** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgets.MenuList	*Clase **MenuList** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgetsTexto.MenuPopup	*Clase **MenuPopup** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgets.MenuPopup	*Clase **MenuPopup** implementa la interfaz widget y hereda los métodos de WidgetParent*

interfaz.Parser	*La clase **Parser** realiza la construcción de la interfaz grafica a partir de un archivo XUL*
interfaz.ParserTexto	*La clase **ParserTexto** realiza la construcción de la interfaz grafica en modo texto a partir de un archivo XUL*
widgetsTexto.RadioButton	*Clase **RadioButton** implementa la interfaz Widget*
widgets.RadioButton	*Clase **RadioButton** implementa la interfaz Widget*
widgetsTexto.RadioGroup	*Clase **RadioGroup** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgets.RadioGroup	*Clase **RadioGroup** implementa la interfaz widget y hereda los métodos de WidgetParent*
interfaz.Reference	*Esta clase devuelve la referencia a un objeto identificado por el id object_name*
interfaz.SwingEngine	*La Clase **SwingEngine** realiza la generación de cada uno de los elementos de la interfaz en modo swing*
interfaz.SwingTextTutorial	*Esta clase contiene una pequeña aplicación que muestra los diferentes widgets implementados por cada tipo de interfaz*
widgetsTexto.TextBox	*Clase **TextBox** implementa la interfaz Widget*
widgets.TextBox	*Clase **TextBox** implementa la interfaz Widget*
interfaz.TextEngine	*La Clase **TextEngine** realiza la generación de cada uno de los elementos de la interfaz en modo texto*
widgets.Vbox	*Clase **Vbox** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgetsTexto.Vbox	
GenericElements.widget	
GenericElements.WidgetChild	*La clase **WidgetChild** implementa la interfaz widget para elementos no contenedores en la interfaz tipo swing*
GenericElements.WidgetChildTexto	*La clase **WidgetChildTexto** implementa la interfaz widget para elementos no contenedores en la interfaz tipo texto*
GenericElements.WidgetParent	*La clase WidgetParemt implementa la interfaz widget para elementos contenedores de tipo swing*
GenericElements.WidgetParentTexto	*La clase **WidgetChild** implementa la interfaz widget para elementos no contenedores en la interfaz tipo texto*
widgets.WindowBox	*Clase **WindowBox** implementa la interfaz widget y hereda los métodos de WidgetParent*
widgetsTexto.WindowBox	*Clase **WindowBox** implementa la interfaz widget y hereda los métodos de WidgetParent*

Estructura de directorios del proyecto OSGi-UI

- **jbproject**
  - **OSGi-UI_V1_0**
    - **src**
      - **GenericElements**
      - **interfaz**
      - **utils**
      - **widgets**
      - **widgetsTexto**

Lista de ficheros del proyecto OSGi-UI

jbproject/OSGi-UI_V1_0/src/GenericElements/GenericEngine.java	
jbproject/OSGi-UI_V1_0/src/GenericElements/GenericParser.java	
jbproject/OSGi-UI_V1_0/src/GenericElements/widget.java	
jbproject/OSGi-UI_V1_0/src/GenericElements/WidgetChild.java	
jbproject/OSGi-UI_V1_0/src/GenericElements/WidgetChildTexto.java	
jbproject/OSGi-UI_V1_0/src/GenericElements/WidgetParent.java	
jbproject/OSGi-UI_V1_0/src/GenericElements/WidgetParentTexto.java	
jbproject/OSGi-UI_V1_0/src/interfaz/Error.java	
jbproject/OSGi-UI_V1_0/src/interfaz/EventHandlers.java	
jbproject/OSGi-UI_V1_0/src/interfaz/EventThrowing.java	
jbproject/OSGi-UI_V1_0/src/interfaz/Parser.java	
jbproject/OSGi-UI_V1_0/src/interfaz/ParserTexto.java	
jbproject/OSGi-UI_V1_0/src/interfaz/Reference.java	
jbproject/OSGi-UI_V1_0/src/interfaz/SwingEngine.java	
jbproject/OSGi-UI_V1_0/src/interfaz/SwingTextTutorial.java	
jbproject/OSGi-UI_V1_0/src/interfaz/TextEngine.java	
jbproject/OSGi-UI_V1_0/src/utils/ImagePanel.java	
jbproject/OSGi-UI_V1_0/src/utils/IntegerID.java	
jbproject/OSGi-UI_V1_0/src/utils/JDomUtils.java	
jbproject/OSGi-UI_V1_0/src/widgets/BoxFiller.java	
jbproject/OSGi-UI_V1_0/src/widgets/Button.java	
jbproject/OSGi-UI_V1_0/src/widgets/Hbox.java	
jbproject/OSGi-UI_V1_0/src/widgets/Images.java	
jbproject/OSGi-UI_V1_0/src/widgets/Label.java	
jbproject/OSGi-UI_V1_0/src/widgets/Listbox.java	
jbproject/OSGi-UI_V1_0/src/widgets/ListItem.java	
jbproject/OSGi-UI_V1_0/src/widgets/MenuItem.java	
jbproject/OSGi-UI_V1_0/src/widgets/MenuList.java	
jbproject/OSGi-UI_V1_0/src/widgets/MenuPopup.java	
jbproject/OSGi-UI_V1_0/src/widgets/RadioButton.java	
jbproject/OSGi-UI_V1_0/src/widgets/RadioGroup.java	
jbproject/OSGi-UI_V1_0/src/widgets/TextBox.java	
jbproject/OSGi-UI_V1_0/src/widgets/Vbox.java	
jbproject/OSGi-UI_V1_0/src/widgets/WindowBox.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/Button.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/Hbox.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/Label.java	

jbproject/OSGi-UI_V1_0/src/widgetsTexto/Listbox.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/ListItem.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/MenuItem.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/MenuList.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/MenuPopup.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/RadioButton.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/RadioGroup.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/TextBox.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/Vbox.java	
jbproject/OSGi-UI_V1_0/src/widgetsTexto/WindowBox.java	