

Coulomb Nicolas  
Gasiglia Baptiste

Licence Professionnelle  
Des Métiers de l'Informatique

# Annexes

## Du rapport OFL-Meta

*Responsables du projet :*

M. Pierre Crescenzo

M. Philippe Lahire

Le 21/01/02

## Index des annexes

Annexe 1 :	La classe Fenetre.....	P.3
Annexe 2 :	La classe MenuFile.....	P.6
Annexe 3 :	La classe JStdTools.....	P.13
Annexe 4 :	La classe RightPanel.....	P.15
Annexe 5 :	La classe LeftTree.....	P.18
Annexe 6 :	La classe MenuPopup.....	P.20
Annexe 7 :	La classe Nodecomponent.....	P.25
Annexe 8 :	La classe PopupListener.....	P.44
Annexe 9 :	La classe ConstantesString.....	P.47
Annexe 10 :	La classe OFL-Meta.....	P.50

## Annexe 1 : La classe Fenetre

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;
import javax.swing.UIManager;
import javax.swing.JScrollPane;
import java.awt.event.MouseAdapter;

/**
 * This is the principal class of the OFL Méta software.
 * This class extend from the JFrame class and contain
 * all the panel and the contener of the application.
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public class Fenetre extends JFrame implements ConstantesString
{
    /*
     * This is the library of names using during class Fenetre.
     */
    //String exitMsgBoxText= "A bientot sur OFL-META";

    /*
     * Set the dimension of the principal frame
     */
    private Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    private int VSIZE=screenSize.height-40;
    private int HSIZE=screenSize.width;

    /*
     * This is teh local variable
     */
    private int splitDefaultSize = 250;
    /*private String SoftIcon= "Help.gif";
    private String statusBarText="For Help Press F1";*/

    /**
     * @see LeftTree#LeftTree(RightPanel)
     */
    private RightPanel jRightPanel = new RightPanel();
    private LeftTree jTree = new LeftTree(jRightPanel);
    private MenuFile jMenuFile = new MenuFile(jTree.jTreeLanguage);

    /*
     * This is where we're create our different panel
     */
    private JLabel statusBar = new JLabel();
    private JSplitPane jScreenSeparator = new JSplitPane();
    private JOptionPane jPaneQuit = new JOptionPane();
    private JScrollPane scrollerLeft = new JScrollPane(jTree.jTreeLanguage);
```

```
private JScrollPane scrollerRight = new
JScrollPane(jRightPanel,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,JScrollPane.HORIZONTAL_S
CROLLBAR_AS_NEEDED);

/**
 * @see JStdTools#JStdTools()
 */
public JStdTools stTools = new JStdTools();

/**
 * Class constructor, without parameters
 *
 * @since JDK 1
 */
public Fenetre()
{
    /*
     * This is the integration of the constructor
     */
    setSize(HSIZE,VSIZE);
    setTitle("OFL-META Version 1.0a");
    setJMenuBar(jMenuFile);
    statusBar.setText(statusBarText);

    /*
     * The frame closed control
     */
    addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent e)
        {
            JLabel component = new JLabel(exitMsgBoxText);
            JOptionPane.showMessageDialog(null , component , "Au revoir"
,JOptionPane.INFORMATION_MESSAGE );
            System.exit(0);
        }
    });

    /*
     * The right clic control
     */
    addMouseListener(new MouseAdapter()
    {
        public void mouseClicked(MouseEvent e)
        {
            //System.out.println(e);
        }
    });

    /*
     * We using a toolkit to put an icon associate to the soft
     */
    Toolkit icone = Toolkit.getDefaultToolkit();
    Image JFrameIcon = icone.getImage(SoftIcon);
    setIconImage(JFrameIcon);

    /*
     * This is where we creat the drawn work panel
     */
    Container containPanel = getContentPane();
```

```
// Creation des cadres autour des deux fenetres
//jLeftPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(),"Arbre"
));
//jRightPanel.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(),"Texte"
));

/*
 * This is the creation of all the split séparator of OFL Méta interface
 */
jScreenSeparator.add(scrollerRight/*jRightPanel*/, JSplitPane.RIGHT );
jScreenSeparator.add(scrollerLeft/*jTree.jTreeLanguage*/, JSplitPane.LEFT );
jScreenSeparator.setDividerLocation(splitDefaultSize);

scrollerLeft.setBackground(Color.white);

/*
 * This is where we add the panel and the scroll
 */
containPanel.add(jScreenSeparator, "Center");
containPanel.add(statusBar, "South");

/*
 * To see OFL Méta interface
 */
setVisible(true);
}
}
```

## Annexe 2 : La classe MenuFile

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;
import javax.swing.JOptionPane;

/**
 * This class "nous permet" to initialise all the menu
 * We also describe the actionPerformed which are use on
 * the Menu method
 * <br>
 * This class extend form the JToolBar class
 * and implements the ActionListener class
 * <br>
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public class MenuFile extends JMenuBar implements ActionListener, ConstantesString
{

    private JTree languageTree;
    private DefaultTreeModel languageModel;
    private RightPanel theRightPanel;

    /*
     * This are the local variable
     */
    private JMenuItem menuApropos, menuHelp;
    private JMenuItem menuNew, menuOpen, menuImport, menuSaveAs, menuSave, menuQuit;
    private JMenuItem menuAdd, menuRemove, menuRename, menuCopy, menuPaste, menuExpand,
menuCollapse;
    private JTree jTree;
    private NodeComponent referNode;

    /**
     * Class constructor, without parameters
     * <br>
     * Initialize our menu on the top of the OFL Méta
     * interface
     *
     * @since JDK1
     */
    public MenuFile(JTree zeJTree)
    {

        jTree = zeJTree;

        // create tree menu : File, Option and About
        JMenu menuFile = new JMenu(fileText);
        JMenu menuOption = new JMenu(optionText);
```

```
JMenu menuAbout = new JMenu(helpText);

/***** Menu File *****/

/*
 * This is the new buton
 * The icon is New.gif
 * The keyboard ShortCut is ctrl n
 */
menuNew = new JMenuItem(newText);
menuNew.setIcon(new ImageIcon(MenuFile.class.getResource(newIcon)));
menuNew.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_N,
java.awt.Event.CTRL_MASK));

/*
 * This is the open buton
 * The icon is Open.gif
 * The keyboard ShortCut is ctrl o
 */
menuOpen = new JMenuItem(openText);
menuOpen.setIcon(new ImageIcon(MenuFile.class.getResource(openIcon)));
menuOpen.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_O,
java.awt.Event.CTRL_MASK));

/*
 * This is the import buton
 * The keyboard ShortCut is ctrl i
 */
menuImport = new JMenuItem(importText);
menuImport.setIcon(new ImageIcon(MenuFile.class.getResource(openIcon)));
menuImport.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_I,
java.awt.Event.CTRL_MASK));

/*
 * This is the save as buton
 * The icon is Save As.gif
 * The keyboard ShortCut is ctrl d
 */
menuSaveAs = new JMenuItem(saveAsText);
menuSaveAs.setIcon(new ImageIcon(MenuFile.class.getResource(saveAsIcon)));
menuSaveAs.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_D,
java.awt.Event.CTRL_MASK));

/*
 * This is the save buton
 * The icon is Save Save.gif
 * The keyboard ShortCut is ctrl s
 */
menuSave = new JMenuItem(saveText);
menuSave.setIcon(new ImageIcon(MenuFile.class.getResource(saveIcon)));
menuSave.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S,
java.awt.Event.CTRL_MASK));

/*
 * This is the quit buton
 */
menuQuit = new JMenuItem(quitText);
//menuQuit.setIcon(new ImageIcon(MenuFile.class.getResource("Quit.gif"));
```

```
/* Menu Option */

/*
 * This is the add buton
 * The icon is Save Add.gif
 * The keyboard ShortCut is ctrl a
 */
menuAdd = new JMenuItem(addText);
menuAdd.setIcon(new ImageIcon(MenuFile.class.getResource(addIcon)));
menuAdd.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_A,
java.awt.Event.CTRL_MASK));

/*
 * This is the remove buton
 * The icon is Save Remove.gif
 * The keyboard ShortCut is ctrl r
 */
menuRemove = new JMenuItem(removeText);
menuRemove.setIcon(new ImageIcon(MenuFile.class.getResource(removeIcon)));
menuRemove.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_R,
java.awt.Event.CTRL_MASK));

/*
 * This is the Rename buton
 * The keyboard ShortCut is ctrl e
 */
menuRename = new JMenuItem(renameText);
menuRename.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_E,
java.awt.Event.CTRL_MASK));

/*
 * This is the copy buton
 * The keyboard ShortCut is ctrl r
 */
menuCopy = new JMenuItem(copyText);
menuCopy.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_C,
java.awt.Event.CTRL_MASK));

/*
 * This is the paste buton
 * The keyboard ShortCut is ctrl v
 */
menuPaste = new JMenuItem(pasteText);
menuPaste.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_V,
java.awt.Event.CTRL_MASK));

/*
 * This is the expand buton
 * The keyboard ShortCut is ctrl p
 */
menuExpand = new JMenuItem(expandText);
menuExpand.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_P,
java.awt.Event.CTRL_MASK));

/*
 * This is the collapse buton
 * The keyboard ShortCut is ctrl m
 */
menuCollapse = new JMenuItem(collapseText);
```



```
menuCollapse.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_M,  
java.awt.Event.CTRL_MASK));
```

```
/* Menu About */
```

```
/*  
 * This is the a propos buton  
 */  
menuApropos = new JMenuItem(aProposText);
```

```
/*  
 * This is the help buton  
 * The icon is Save Help.gif  
 * The keyboard ShortCut is ctrl F1  
 */  
menuHelp = new JMenuItem(helpContentsText);  
menuHelp.setIcon(new ImageIcon(MenuFile.class.getResource(helpIcon)));  
menuHelp.setAccelerator(KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_F1,  
java.awt.Event.CTRL_MASK));
```

```
/*  
 * "Ajout des contenus dans les menus"  
 */  
menuFile.setMnemonic(fileShortCut); // keyboard ShortCut : alt F  
menuFile.add(menuNew);  
menuFile.add(menuOpen);  
menuFile.add(menuImport);  
//-----  
menuFile.addSeparator();  
menuFile.add(menuSaveAs);  
menuFile.add(menuSave);  
//-----  
menuFile.addSeparator();  
menuFile.add(menuQuit);
```

```
menuOption.setMnemonic(optionShortCut); // keyboard ShortCut : alt O  
menuOption.add(menuAdd);  
menuOption.add(menuRemove);  
//-----  
menuOption.addSeparator();  
menuOption.add(menuAdd);  
menuOption.add(menuRemove);  
menuOption.add(menuRename);  
menuOption.add(menuCopy);  
menuOption.add(menuPaste);  
//-----  
menuOption.addSeparator();  
menuOption.add(menuCollapse);  
menuOption.add(menuExpand);
```

```
menuAbout.setMnemonic(aboutShortCut); // keyboard ShortCut : alt A  
menuAbout.add(menuApropos);  
//-----  
menuAbout.addSeparator();  
menuAbout.add(menuHelp);
```

```
/*  
 * "Mise en place des ActionListener sur chaque bouton"  
 */
```

```
        menuNew.addActionListener(this);
        menuOpen.addActionListener(this);
        menuSave.addActionListener(this);
        menuSaveAs.addActionListener(this);
        menuQuit.addActionListener(this);
        menuAdd.addActionListener(this);
        menuRemove.addActionListener(this);
        menuApropos.addActionListener(this);
        menuHelp.addActionListener(this);
        menuRename.addActionListener(this);
        menuCopy.addActionListener(this);
        menuPaste.addActionListener(this);
        menuExpand.addActionListener(this);
        menuCollapse.addActionListener(this);

        /*
         * "Insertion des menus"
         */
        add(menuFile);
        add(menuOption);
        add(menuAbout);
    }

    public void actionPerformed(ActionEvent evt)
    {

        TreePath currentSelection = jTree.getSelectionPath();

        if(currentSelection != null)// if no node are select
        {
            referNode = (NodeComponent)(currentSelection.getLastPathComponent());
        }

        /*
         * "Gestion des événements liés aux menus"
         */
        String ChoixOption = evt.getActionCommand();

        /*
         * The new button action
         */
        if (ChoixOption.equals(newText))
        {
            System.out.println(newText);
        }

        /*
         * The open button action
         */
        if (ChoixOption.equals(openText))
        {
            System.out.println(openText);
        }

        /*
         * The save button action
         */
        if (ChoixOption.equals(saveText))
        {
```

```
        System.out.println(saveText);
    }

    /*
     * The saveAs button action
     */
    if (ChoixOption.equals(saveAsText))
    {
        System.out.println(saveAsText);
    }

    /*
     * Open au message box when the user quit OFL Méta
     */
    /* The quit button action
     */
    if (ChoixOption.equals(quitText))
    {
        JLabel component = new JLabel(exitMsgBoxText);
        JOptionPane.showMessageDialog(null , component , "Au revoir"
,JOptionPane.INFORMATION_MESSAGE );
        System.exit(0);
    }

    /*
     * The add button action
     */
    if (ChoixOption.equals(addText))
    {
        referNode.add((DefaultTreeModel)jTree.getModel());
    }

    /*
     * The remove button action
     */
    if (ChoixOption.equals(removeText))
    {
        referNode.remove((DefaultTreeModel)jTree.getModel());
    }

    /*
     * The rename button action
     */
    if (ChoixOption.equals(renameText))
    {
        referNode.rename();
    }

    /*
     * The remove button action
     */
    if (ChoixOption.equals(copyText))
    {
        referNode.copy();
    }

    /*
     * The paste button action
     */
    if (ChoixOption.equals(pasteText))
```

```
{
    referNode.paste((DefaultTreeModel)jTree.getModel());
}

/*
 * The collapse button action
 */
if (ChoixOption.equals(collapseText))
{
    referNode.collapse(jTree);
}

/*
 * The expand button action
 */
if (ChoixOption.equals(expandText))
{
    referNode.expand(jTree);
}

/*
 * Open au message box when the user clic on the aPropos buton
 *
 * The aPropos button action
 */
if (ChoixOption.equals(aProposText))
{
    JLabel component = new JLabel( "Developped by .Bapt & KifT" );
    JOptionPane.showMessageDialog(null , component , aProposText
,JOptionPane.INFORMATION_MESSAGE );
}

/*
 * Open au message box when the user clic on the help buton
 *
 * The help button action
 */
if (ChoixOption.equals(helpContentsText))
{
    JLabel component = new JLabel( "If you have a question, please send mail to kift@ifrance.com or
baptiste.gasiglia@clicasso.org" );
    JOptionPane.showMessageDialog(null , component , helpContentsText
,JOptionPane.INFORMATION_MESSAGE );
}
}
```

## Annexe 3 : La classe JStdTools

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;
import javax.swing.UIManager;

/**
 * This class "nous permet" to creat and add
 * "une barre d'outil reprenant les principales fonctionnalités
 * en y insérant de petits icônes
 * <br>
 * This class extend form the JToolBar class
 * <br>
 * We actually don't use this class. In the class Fenetre
 * we don't call the JStdTools constructor
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public class JStdTools extends JToolBar implements ConstantesString
{
    /**
     * Class constructor, without parameters
     *
     * @since JDK1
     */
    public JStdTools()
    {
        /**
         * The New button
         */
        JButton buttonNew = new JButton(new ImageIcon(newIcon));
        buttonNew.setToolTipText(newTTPText);

        /**
         * The Open button
         */
        JButton buttonOpen = new JButton(new ImageIcon(openIcon));
        buttonOpen.setToolTipText(openTTPText);

        /**
         * The Save button
         */
        JButton buttonSave = new JButton(new ImageIcon(saveIcon));
        buttonSave.setToolTipText(saveTTPText);

        /**
         * The SaveAs button
         */
        JButton buttonSaveAs = new JButton(new ImageIcon(saveAsIcon));
        buttonSaveAs.setToolTipText(saveAsTTPText);
    }
}
```

```
        /*
        * The Add button
        */
        JButton buttonAdd = new JButton(new ImageIcon(addIcon));
        buttonAdd.setToolTipText(addTTPText);

        /*
        * The Remove button
        */
        JButton buttonRemove = new JButton(new ImageIcon(removeIcon));
        buttonRemove.setToolTipText(removeTTPText);

        /*
        * This is where we add all button to JStdTools
        */
        add(buttonNew);
        add(buttonOpen);
        add(buttonSave);
        add(buttonSaveAs);
        add(buttonAdd);
        add(buttonRemove);
    }
}
```

## Annexe 4 : La classe RightPanel

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

/**
 * This class extends from the JPanel class
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK 1
 */
public class RightPanel extends JPanel implements ConstantesString
{
    /**
     * Class constructor : Creat a nex Right Panel
     *
     * @since JDK 1
     */
    public RightPanel()
    {
        super();
        setBackground(Color.white);
    }

    /**
     * @post the select node contain directories
     * @pre a new icone of directories is add on the right panel
     *
     * @param JTree the curent tree
     * @since JDK 1
     */
    public void addDir(JTree refTree)
    {
        NodeComponent referNode;
        NodeComponent alpha;
        JLabel RightLabel;
        JLabel nameLabel;

        setBackground(Color.white);
        FlowLayout fLayout= new FlowLayout(FlowLayout.LEADING);
        setLayout(fLayout);

        TreePath currentSelection = refTree.getSelectionPath();

        if(currentSelection != null)
        {
            referNode = (NodeComponent)(currentSelection.getLastPathComponent());

            for (Enumeration enu=referNode.children(); enu.hasMoreElements()!=false;)
            {
```

```
        alpha =(NodeComponent) enu.nextElement();

        if(alpha.children().hasMoreElements()==false)
        {
            nameLabel = new JLabel(alpha.getUserObject().toString());
            RightLabel = new JLabel(new
ImageIcon(RightPanel.class.getResource("File.gif")));
        }
        else
        {
            nameLabel = new JLabel(alpha.getUserObject().toString());
            RightLabel = new JLabel(new
ImageIcon(RightPanel.class.getResource("Repertoire.gif")));
        }

        add(nameLabel);
        add(RightLabel);
    }
    setPreferredSize(new Dimension(getParent().getWidth(),7680));
    updateUI();
}

/**
 *
 * @post the select node contain files
 * @pre a new icone of files is add on the right panel
 *
 * @param JTree the curent tree
 * @since JDK 1
 */
public void addFile(JTree refTree)
{
    NodeComponent referNode;

    setBackground(Color.white);
    FlowLayout fLayout= new FlowLayout(FlowLayout.LEADING);
    setLayout(fLayout);

    TreePath currentSelection = refTree.getSelectionPath();

    if(currentSelection != null)
    {
        referNode = (NodeComponent)(currentSelection.getLastPathComponent());

        JLabel jLab = new JLabel(referNode.toString());
        add(jLab);

        if(referNode.getNameLanguage() == false)
        {
            JTextField jText1 = new JTextField (30);
            add(jText1);
            JTextField jText2 = new JTextField (referNode.getNodeValue(), 30);
            add(jText2);
        }
        else
        {
            JTextField jText1 = new JTextField (60);
            add(jText1);
        }
    }
}
```



```
        }
        updateUI();
    }

/**
 *
 * @post the select node contain files or directories, they are add on the panel
 * @pre the files or the directories are draw with a text
 *
 * @param JTree the curent tree
 * @since JDK 1
 */
public void addSubData(JTree refTree)
{
    NodeComponent referNode;
    NodeComponent alpha;
    JLabel RightLabel;
    JLabel nameLabel;

    setBackground(Color.white);
    FlowLayout fLayout= new FlowLayout(FlowLayout.LEADING);
    setLayout(fLayout);

    TreePath currentSelection = refTree.getSelectionPath();

    if(currentSelection != null)
    {
        referNode = (NodeComponent)(currentSelection.getLastPathComponent());

        for (Enumeration enu=referNode.getChildAt(0).children();
            enu.hasMoreElements()!=false;)
        {
            alpha = (NodeComponent) enu.nextElement();

            JPanel jTempPanel = new JPanel();
            jTempPanel.setBackground(Color.white);

            JLabel jLab = new JLabel(alpha.toString());
            jTempPanel.add(jLab);

            if(alpha.getNameLanguage() == false)
            {
                JTextField jText1 = new JTextField (30);
                jTempPanel.add(jText1);
                JTextField jText2 = new JTextField (alpha.getNodeValue(), 30);
                jTempPanel.add(jText2);
            }
            else
            {
                JTextField jText1 = new JTextField (60);
                jTempPanel.add(jText1);
            }

            add(jTempPanel);
        }
    }
    setPreferredSize(new Dimension(getParent().getWidth(),7680));
    System.out.println(getHeight());
    updateUI();
}
```

}

## Annexe 5 : La classe LeftTree

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

/**
 * In this class we only creat the "base" of the tree
 * that wee can see when we open OFL Méta
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public class LeftTree implements ConstantesString
{
    /**
     * We declare the save path right panel
     */
    RightPanel zeRightPanel;

    /**
     * The creation of the root node
     *
     * @see NodeComponent#NodeComponent(String, String)
     */
    private NodeComponent rootNode = new NodeComponent(defaultTreeText,null);

    /**
     * The creation of the type of the tree
     */
    private DefaultTreeModel treeModel = new DefaultTreeModel(rootNode);

    /**
     * The creation of the Jtree
     */
    public JTree jTreeLanguage = new JTree(treeModel);

    /**
     * Class constructor,spécifing the "chemin" of the panel
     * where the tree will be creat.
     *
     * @since JDK1.3
     */
    public LeftTree(RightPanel jRightPanel)
    {
        zeRightPanel = jRightPanel;

        /**
         * To "rendre" not editable all the language
         */
    }
}
```

```
* @see NodeComponent#setEditable(boolean)
*/
    jTreeLanguage.setEditable(false);

    /*
    * To see the root node
    */

jTreeLanguage.getSelectionModel().setSelectionMode(TreeSelectionMode.SINGLE_TREE_SELECTION);

/*
    * ***** A expliquer *****
    */
jTreeLanguage.setShowsRootHandles(true);

/***** Add two directory at level 0 *****/

/**
 * @see NodeComponent#setAllFalse()
 */
rootNode.setAllFalse();

/**
 * @see NodeComponent#addLanguage(String, DefaultTreeModel, boolean)
 */
rootNode.addLanguage("Current Language", treeModel, true);

    /**
    * This is the libraryNode where is in the rootNode
    *
    * @see NodeComponent#NodeComponent(String, String)
    */
    NodeComponent libraryNode = new NodeComponent(treeLibrary,null);
    rootNode.add(libraryNode, treeModel);

/**
 * @see NodeComponent#setAllFalse()
 */
libraryNode.setAllFalse();

/***** Add two directory at level 1 *****/

/**
 * @see NodeComponent#addLanguage(String, DefaultTreeModel, boolean)
 */
libraryNode.addLanguage(nameLibrary1, treeModel, false);
libraryNode.addLanguage(nameLibrary2, treeModel, false);

/**
 * tree listener (see class PopupListener)
 *
 * @see PopupListener#PopupListener(JTree, DefaultTreeModel, RightPanel)
 */
    PopupListener pl = new PopupListener(jTreeLanguage,treeModel,zeRightPanel);
    jTreeLanguage.addMouseListener(pl);
}
}
```

## Annexe 6 : La classe MenuPopup

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

/**
 * This class creat the 'affichage" of a popupmenu
 * which "apparait" when the user clic with his right
 * buton on a tree element
 * <br>
 * This class extend form the JPopupMenu class
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK 1
 */
public class MenuPopup extends JPopupMenu implements ConstantesString
{
    /*
     *
     */
    private JMenuItem NewMenu;

    /**
     * Class constructor, build the popup menu din the
     * "refTree" which model is "lgModel" and when we
     * clic on the "ref" node
     * <br>
     * All the parameters are declared like "final"
     * because this parameters have to be modified
     *
     * @param ref "la node de référence"
     * @param lgModel "le model de l'arbre"
     * @param refTree "la référence de l'arbre"
     * @since JDK 1
     */
    public MenuPopup(final NodeComponent ref, final DefaultTreeModel lgModel, final JTree refTree)
    {
        /*
         * This is the declaration of the ActionListener
         * "qui gère les évènements sur le popup menu"
         */
        ActionListener al = new ActionListener()
        {
            /**
             * @param actionPerformed The select evnement
             * @since JDK 1
             */
            public void actionPerformed(ActionEvent e)
            {
                /*
```

```
* If the user select the add fonction
*/
if(((JMenuItem)e.getSource()).getText() == addText)
{
    /**
     * @see NodeComponent#add(DefaultTreeModel)
     */
    ref.add(lgModel);
    //refTree.updateUI();
}

/*
 * If the user select the remove fonction
 */
if(((JMenuItem)e.getSource()).getText() == removeText)
{
    /**
     * @see NodeComponent#remove(DefaultTreeModel)
     */
    ref.remove(lgModel);
    //refTree.updateUI();
}

/*
 * If the user select the rename fonction
 */
if(((JMenuItem)e.getSource()).getText() == renameText)
{
    /**
     * "Windows update"
     */
    /**
     * @see NodeComponent#rename()
     */
    ref.rename();
    refTree.updateUI();
}

/*
 * If the user select the copy fonction
 */
if(((JMenuItem)e.getSource()).getText() == copyText)
{
    /**
     * "Windows update"
     */
    /**
     * @see NodeComponent#copy()
     */
    ref.copy();
    refTree.updateUI();
}

/*
 * If the user select the paste fonction
 */
if(((JMenuItem)e.getSource()).getText() == pasteText)
{
    /**
     * "Windows update"
     */
    /**
     * @see NodeComponent#paste(DefaultTreeModel)
     */

```

```
        */
        ref.paste(lgModel);
        refTree.updateUI();
    }

    /*
     * If the user select the expand fonction
     */
    if(((JMenuItem)e.getSource()).getText() == expandText)
    {
        /**
         * "Windows update"
         *
         * @see NodeComponent#expand(DefaultTreeModel)
         */
        ref.expand(refTree);
        refTree.updateUI();
    }

    /*
     * If the user select the collapse fonction
     */
    if(((JMenuItem)e.getSource()).getText() == collapseText)
    {
        /**
         * "Windows update"
         *
         * @see NodeComponent#collapse(JTree)
         */
        ref.collapse(refTree);
        refTree.updateUI();
    }
}
};

/*
 * Insert fonction add into default popup menu
 */

NewMenu = new JMenuItem(addText);
NewMenu.setIcon(new ImageIcon(MenuFile.class.getResource(addIcon)));
NewMenu.addActionListener(al);
add(NewMenu);

/**
 * @see NodeComponent#isAddable
 */
if(!ref.getAddable())
{
    NewMenu.setEnabled(false);
}

/*
 * Insert fonction remove into default popup menu
 */

NewMenu = new JMenuItem(removeText);
NewMenu.setIcon(new ImageIcon(MenuFile.class.getResource(removeIcon)));
NewMenu.addActionListener(al);
add(NewMenu);
```

```
/**
 * @see NodeComponent#isRemovable
 */
if(!ref.getRemovable())
{
    NewMenu.setEnabled(false);
}

/*
 * Insert function rename into default popup menu
 */
NewMenu = new JMenuItem(renameText);
NewMenu.addActionListener(al);
add(NewMenu);

if(!ref.getModifiable())
{
    NewMenu.setEnabled(false);
}

//-----
addSeparator();
//-----

/*
 * Insert function copy into default popup menu
 */
NewMenu = new JMenuItem(copyText);
NewMenu.addActionListener(al);
add(NewMenu);

/**
 * @see NodeComponent#isCopiable
 */
if(!ref.getCopiable())
{
    NewMenu.setEnabled(false);
}

/*
 * Insert function paste into default popup menu
 */
NewMenu = new JMenuItem(pasteText);
NewMenu.addActionListener(al);
add(NewMenu);

/**
 * @see NodeComponent#isAddable
 * @see NodeComponent#getType()
 * @see NodeComponent#nodeSaved this is a staitc atribute
 */
if(ref.getNodeSaved() == null || !ref.getAddable() || ref.getType() !=
ref.getNodeSaved().getType())
{
    NewMenu.setEnabled(false);
}

//-----
addSeparator();
//-----
```

```
        /*
    * Insert fonction colpaste into default popup menu
    */
        NewMenu = new JMenuItem(collapseText);
        NewMenu.addActionListener(al);
        add(NewMenu);

        /*
    * Insert fonction expand into default popup menu
    */
        NewMenu = new JMenuItem(expandText);
        NewMenu.addActionListener(al);
        add(NewMenu);
    }
}
```



## Annexe 7 : La classe NodeComponent

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

/**
 *
 * This class extends fromm the DefaultMutableTreeNode class
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK 1
 */
public class NodeComponent extends DefaultMutableTreeNode implements ConstantesString
{
    /**
     * Give the number of the last new node
     */
    private static int newNodeSuffix = 1;

    /**
     * Give the type of the parent node (description component, language component, relationship
    component)
     */
    private String type;

    /**
     * All this boolean allow fonction to be used
     * This are all true "par defaut" except isNameLanguage
     * wich is false
     */
    private boolean isNameLanguage = false;
    private boolean isModifiable = true;
    private boolean isAddable = true;
    private boolean isRemovable = true;
    private boolean isCopiable = true;

    /**
     * The node which is save for the copy fonction
     *
     * @see NodeComponent#copy()
     */
    private static NodeComponent nodeSaved;

    /**
     * Information about the node "(ou des option du composant) sera deplacé dans cmpDesc"
     */
    private String nodeValue;

    //ComponentDescription cDes;
}
```

```
/**
 * Class constructor : it creat a new node
 *
 * @post creat a new node with the type gived in parameter
 * @pre theType is a String attribut
 *
 * @param theType the type of the node
 * @since JDK 1
 */
public NodeComponent(String theType)
{
    super("Node " + newNodeSuffix);
    type=theType;
    /*cDes = new ComponentDescription();*/
    newNodeSuffix++;
    setAddable(false);
}

/**
 * Class constructor : it creat a new node
 *
 * @post creat a new node with the name and the type gived in parameter
 * @pre name, value and theType are some String attribut
 *
 * @param name the name of the node
 * @param theType the type of the node
 * @since JDK 1
 */
public NodeComponent(String name, String theType)
{
    super(name);
    type=theType;
    // cDes = new ComponentDescription();
}

/**
 * Class constructor : it creat a new node with a value
 *
 * @post creat a new node with the name, type and value gived in parameter
 * @pre name, value and theType are some String attribut
 *
 * @param name the name of the node
 * @param value the value of the node
 * @param theType the type of the node
 * @since JDK 1
 */
public NodeComponent(String name, String value, String theType)
{
    super(name);
    type=theType;
    nodeValue=value;
    // cDes = new ComponentDescription();
}

/**
 *
 * @post the (parameter) Node is add to the (parameter) tree
 * @pre Node is a NodeComponent attribut and tree is a DefaultTreeModel attribut
 *
```

```
    * @param Node the node thta we want to add
    * @param tree the tree where we want to add the node
    * @since JDK 1
    */
public void add(NodeComponent Node, DefaultTreeModel tree)
{
    /*
    * add new node in curent selection or to the parent node
    */
    tree.insertNodeInto(Node, this,this.getChildCount());
}

/**
 * Add e new node
 *
 * @post a new node is add to the tree gived in parameter
 * @pre tree is a DefaultTreeModel attribut
 *
 * @param tree the tree where we want to add the node
 * @since JDK 1
 */
public void add(DefaultTreeModel tree)
{
    NodeComponent addedNode = new NodeComponent(getType());
    // add new node in curent selection or to the parent node
    tree.insertNodeInto(addedNode, this,this.getChildCount());
}

/**
 * Remove a node
 *
 * @post remove the select node of the tree gived in parameter
 * @pre tree is a DefaultTreeModel attribut
 *
 * @param tree the tree where we want to remove the node
 * @since JDK 1
 */
public void remove(DefaultTreeModel tree)
{
    tree.removeNodeFromParent(this);
}

/**
 * Set the add caractere of a node
 *
 * @post if boolToAdd is true, node can be add to the select node.
 *         if boolToAdd is false, node can't be add to the select node
 * @pre boolToAdd is a boolean attribut
 *
 * @param boolToAdd true or false
 * @since JDK 1
 */
public void setAddable(boolean boolToAdd)
{
    isAddable=boolToAdd;
}

/**
 * Get the add caractere of a node
 *
```

```

    * @post if boolToAdd is true, node can be add to the select node.
    *           if boolToAdd is false, node can't be add to the select node
    * @pre NodeComponent
    *
    * @return boolean the value of the isAddable attribut
    * @since JDK 1
    */
public boolean getAddable()
{
    return this.isAddable;
}

/**
 * Set the remove caractere of a node
 *
 * @post if boolToRem is true, the select node can be remove
 *           if boolToRem is false, the select node can't be remove
 * @pre boolToRem is a boolean attribut
 *
 * @param boolToRem true or false
 * @since JDK 1
 */
public void setRemovable(boolean boolToRem)
{
    isRemovable=boolToRem;
}

/**
 * Get the remove caractere of a node
 *
 * @post if boolToRem is true, node can be remove to the select node.
 *           if boolToRem is false, node can't be remove to the select node
 * @pre NodeComponent
 *
 * @return boolean the value of the isRemovable attribut
 * @since JDK 1
 */
public boolean getRemovable()
{
    return this.isRemovable;
}

/**
 * Set the modifiable caractere of a node
 *
 * @post if boolToMod is true, the select node can be modifiante
 *           if boolToMod is false, the select node can't be modifiante
 * @pre boolToMod is a boolean attribut
 *
 * @param boolToMod true or false
 * @since JDK 1
 */
public void setModifiable(boolean boolToMod)
{
    isModifiable=boolToMod;
}

/**
 * Get the modif caractere of a node
 *
```

```

    * @post if boolToMod is true, node can be modif to the select node.
    *         if boolToMod is false, node can't be modif to the select node
    * @pre NodeComponent
    *
    * @return boolean the value of the isModifiable attribut
    * @since JDK 1
    */
public boolean getModifiable()
{
    return this.isModifiable;
}

/**
 * Set the copiable caractere of a node
 *
 * @post if boolToCop is true, the select node can be copy
 *         if boolToCop is false, the select node can't be copy
 * @pre boolToCop is a boolean attribut
 *
 * @param boolToCop true or false
 * @since JDK 1
 */
public void setCopiable(boolean boolToCop)
{
    isCopiable=boolToCop;
}

/**
 * Get the copy caractere of a node
 *
 * @post if boolToCop is true, node can be copy to the select node.
 *         if boolToCop is false, node can't be copy to the select node
 * @pre NodeComponent
 *
 * @return boolean the value of the isCopiable attribut
 * @since JDK 1
 */
public boolean getCopiable()
{
    return this.isCopiable;
}

/**
 * Set the Addable, Removable, Copiable and Modifiable caractere of a node
 *
 * @post the node isn't Addable, Removable, Copiable and Modifiable
 * @pre the node exist
 *
 * @since JDK 1
 */
public void setAllFalse()
{
    this.setAddable(false);
    this.setRemovable(false);
    this.setCopiable(false);
    this.setModifiable(false);
}

/**
```

```

    * Determine if a node is a name language
    *
    * @post if boolToNam is true, the select node is a name language
    *       if boolToNam is false, the select node isn't a name language
    * @pre boolToNam is a boolean attribut
    *
    * @param boolToNam true or false
    * @since JDK 1
    */
public void setIsNameLanguage(boolean boolToNam)
{
    isNameLanguage=boolToNam;
}

/**
 * Get the isNameLanguage caractere of a node
 *
 * @post if boolToRem is true, the node is a name language node.
 *       if boolToRem is false, the node isn't a name language node
 * @pre NodeComponent
 *
 * @return boolean the value of the isNameLanguage attribut
 * @since JDK 1
 */
public boolean getNameLanguage()
{
    return this.isNameLanguage;
}

/**
 * Set the nodeValue of the node
 *
 * @post
 * @pre
 *
 * @param String the value of the node
 * @since JDK 1
 */
public void setNodeValue(String theValue)
{
    this.nodeValue = theValue;
}

/**
 * Get the nodeValue of the node
 *
 * @post
 * @pre NodeComponent
 *
 * @return String the value of the node
 * @since JDK 1
 */
public String getNodeValue()
{
    return this.nodeValue;
}

/**
 * Get the nodeSaved
 *
```

```
    * @post return the node
    * @pre a node was save
    *
    * @return NodeComponent the node which was saved
    * @since JDK 1
    */
public NodeComponent getNodeSaved()
{
    return this.nodeSaved;
}

/**
 * Expand a node
 *
 * @post expand the select node
 * @pre tree is a JTree attribut
 *
 * @param tree the curent tree
 * @since JDK 1
 */
public void expand(JTree tree)
{
    TreePath newPath = tree.getSelectionPath();
    tree.expandPath(newPath);
}

/**
 * Collapse a node
 *
 * @post Collapse the select node
 * @pre tree is a JTree attribut
 *
 * @param tree the curent tree
 * @since JDK 1
 */
public void collapse(JTree tree)
{
    TreePath newPath = tree.getSelectionPath();
    tree.collapsePath(newPath);
}

/**
 * Copy a node (we do a clone of the node)
 *
 * @post the select node is copy
 * @pre a node is select
 *
 * @since JDK 1
 */
public void copy()
{
    nodeSaved = (NodeComponent) clone();

    /**
     * @see NodeComponent#setRemovable(boolean)
     */
    nodeSaved.setRemovable(true);

    /**
     * @see NodeComponent#setModifiable(boolean)

```

```
        */
        nodeSaved.setModifiable(true);
    }

    /**
     * Add the node save in the clone
     *
     * @post the clone (node) is add to the select node
     * @pre the select node must be addable and the tree is a DefaultTreeModel attribut
     *
     * @param tree the curent tree
     * @since JDK 1
     */
    public void paste(DefaultTreeModel tree)
    {
        tree.insertNodeInto(nodeSaved, this, this.getChildCount());
        nodeSaved=null;
    }

    /**
     * Rename a node and open a "boite de dialogue" to enter the new name
     *
     * @post the select node is rename by the user text enter
     * @pre a node must be select
     *
     * @since JDK 1
     */
    public void rename()
    {
        String msg="Please input a new name for " + getUserObject();
        String inputValue = JOptionPane.showInputDialog(msg);

        if(inputValue !=null && inputValue != "")
        {
            this.setUserObject(inputValue);
        }
    }

    /**
     * Rename a node with the parameter name
     *
     * @post the node is rename
     * @pre the name parameters exist and newName is a String attribut
     *
     * @param String the new name of the node
     * @since JDK 1
     */
    public void rename(String newName)
    {
        this.setUserObject(newName);
    }

    /**
     * To get the type of a node
     *
     * @post we obtain the type of the select node
     * @pre a node is select
     *
     * @return String the type of the select node
     * @since JDK 1
     */
```



```
    */
public String getType()
{
    return this.type;
}

/**
 * Add a new language ("met en place toutes les caractéristiques d'un langage") to the curent tree
 *
 * @post a new language is creat in the current tree
 * @pre nameLanguage is a String attribut, treeModel is a
 *       DefaultTreeModel attribut and value is a boolean attribut
 *
 * @param nameLanguage the name of the language we want creat
 * @param treeModel the tree where the language will be creat
 * @param value the value of the language we want creat
 * @since JDK 1
 */
public void addLanguage(String nameLanguage, DefaultTreeModel treeModel, boolean value)
{

    // Il faut créer une class language qui va créer le langage !!
    // Cette class sera applee ici !!

    /**
     * Add some directory at level 0
     */

    // The ComponentLanguageNode in LanguageNode

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
    NodeComponent rootLanguageNode = new NodeComponent(nameLanguage,null);

    /**
     * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
     */
    this.add(rootLanguageNode, treeModel);

    /**
     * We set the caracter of the node
     *
     * @see NodeComponent#setAllFalse()
     */
    rootLanguageNode.setAllFalse();

    /**
     * Add some directory at level 1
     */

    // The ComponentLanguageNode in LanguageNode

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
}
```

```
        */
    NodeComponent ComponentLanguageNode = new NodeComponent(ComponentLanguageText,
lcText);

    /**
    * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
    */
    rootLanguageNode.add(ComponentLanguageNode, treeModel);

    /**
    * We set the character of the node
    *
    * @see NodeComponent#setAllFalse()
    */
    ComponentLanguageNode.setAllFalse();

    // The ComponentDescriptionNode in LanguageNode

    /**
    * @see NodeComponent#NodeComponent(String, String)
    */
    NodeComponent ComponentDescriptionNode = new NodeComponent(ComponentDescriptionText,
dcText);

    /**
    * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
    */
    rootLanguageNode.add(ComponentDescriptionNode, treeModel);

    /**
    * We set the character of the node
    *
    * @see NodeComponent#setAllFalse()
    */
    ComponentDescriptionNode.setAllFalse();

    // The ComponentRelationNode in LanguageNode

    /**
    * @see NodeComponent#NodeComponent(String, String)
    */
    NodeComponent ComponentRelationNode = new NodeComponent(ComponentRelationText, rcText);

    /**
    * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
    */
    rootLanguageNode.add(ComponentRelationNode, treeModel);

    /**
    * We set the character of the node
    *
    * @see NodeComponent#setAllFalse()
    */
    ComponentRelationNode.setAllFalse();

    /**
    //*****
    //
    Add some directory at level 2
    */
```

```

//*****

// The parameters in ComponentLanguageNode

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
    NodeComponent nameLanguageNode = new NodeComponent(nameText,
ComponentLanguageNode.getType());

    /**
     * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
     */
    ComponentLanguageNode.add(nameLanguageNode, treeModel);

    /**
     * We set the character of the node
     *
     * @see NodeComponent#setAllFalse()
     */
    nameLanguageNode.setAllFalse();

// The parameters in ComponentLanguageNode

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
    NodeComponent classNode = new NodeComponent(classText,
ComponentDescriptionNode.getType());

    /**
     * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
     */
    ComponentDescriptionNode.add(classNode, treeModel);

    /**
     * We set the character of the node
     *
     * @see NodeComponent#setAllFalse()
     */
    classNode.setAllFalse();

// The parameters in ComponentLanguageNode

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
    NodeComponent interfaceNode = new NodeComponent(interfaceText,
ComponentDescriptionNode.getType());

    /**
     * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
     */
    ComponentDescriptionNode.add(interfaceNode, treeModel);

    /**
     * We set the character of the node
     *

```

```
* @see NodeComponent#setAllFalse()
*/
interfaceNode.setAllFalse();

// The parameters in ComponentLanguageNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent importNode = new NodeComponent(importText,
ComponentRelationNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
ComponentRelationNode.add(importNode, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
importNode.setAllFalse();

// The parameters in ComponentLanguageNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent useNode = new NodeComponent(useText, ComponentRelationNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
ComponentRelationNode.add(useNode, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
useNode.setAllFalse();

//*****
//
//                               Add some directory at level 3
//*****

// The parameters in parametersComponentLanguageNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent parametersComponentLanguageNode= new NodeComponent(parametersText,
nameLanguageNode.getType());

/**
```

```
* @see NodeComponent#add(NodeComponent, DefaultTreeModel)
*/
nameLanguageNode.add(parametersComponentLanguageNode, treeModel);

/**
 * We set the character of the node
 */
* @see NodeComponent#setAllFalse()
*/
parametersComponentLanguageNode.setAllFalse();

// The parameters in parametersInterfaceNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent parametersInterfaceNode = new NodeComponent(parametersText,
interfaceNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
interfaceNode.add(parametersInterfaceNode, treeModel);

/*
 * We set the character of the node
 */
parametersInterfaceNode.setAddable(value);
parametersInterfaceNode.setRemovable(false);
parametersInterfaceNode.setCopiable(false);
parametersInterfaceNode.setModifiable(false);

// The parameters in parametersClassNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent parametersClassNode = new NodeComponent(parametersText, classNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
classNode.add(parametersClassNode, treeModel);

/*
 * We set the character of the node
 */
parametersClassNode.setAddable(value);
parametersClassNode.setRemovable(false);
parametersClassNode.setCopiable(false);
parametersClassNode.setModifiable(false);

// The parameters in extendsNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent extendsNode = new NodeComponent(extendsText, importNode.getType());
```

```
/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
importNode.add(extendsNode, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
extendsNode.setAllFalse();

// The parameters in implementNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent implementNode = new NodeComponent(implementText, importNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
importNode.add(implementNode, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
implementNode.setAllFalse();

// The parameters in agregatesNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent agregatesNode = new NodeComponent(agregatesText, useNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
useNode.add(agregatesNode, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
agregatesNode.setAllFalse();

// The parameters in composesNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent composesNode = new NodeComponent(composesText, useNode.getType());
```

```
/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
useNode.add(composesNode, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
composesNode.setAllFalse();

//*****
//
//                               Add some directory at level 4
//*****

    /**
     * Here we Create the nodeLanguageName
     */

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
    NodeComponent nodeLanguageName = new NodeComponent(nameText, nameLanguage,
parametersComponentLanguageNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
parametersComponentLanguageNode.add(nodeLanguageName, treeModel);

/**
 * We set the character of the node
 */
nodeLanguageName.setIsNameLanguage(true);
nodeLanguageName.setAddable(false);
nodeLanguageName.setRemovable(false);
nodeLanguageName.setModifiable(false);

//-----

/**
     * Here we Create the nodeClassName
     */

    /**
     * @see NodeComponent#NodeComponent(String, String)
     */
    NodeComponent nodeClassName = new NodeComponent(nameText,null,
parametersComponentLanguageNode.getType());
    NodeComponent nodeFeatures = new NodeComponent(featuresText, null,
parametersComponentLanguageNode.getType());
    NodeComponent nodeGeneric = new NodeComponent(genericText, null,
parametersComponentLanguageNode.getType());
    NodeComponent nodeFormalGenericTypes = new NodeComponent(formalGenericTypesText, null,
parametersComponentLanguageNode.getType());
    NodeComponent nodeEffectivesTypes = new NodeComponent(effectivesTypesText, null,
parametersComponentLanguageNode.getType());
```

```
/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
parametersClassNode.add(nodeClassName, treeModel);
parametersClassNode.add(nodeFeatures, treeModel);
parametersClassNode.add(nodeGeneric, treeModel);
parametersClassNode.add(nodeFormalGenericTypes, treeModel);
parametersClassNode.add(nodeEffectivesTypes, treeModel);

/*
 * We set the character of the node
 */
/**
 * @see NodeComponent#setAllFalse()
 */
nodeClassName.setAllFalse();
nodeFeatures.setAllFalse();
nodeGeneric.setAllFalse();
nodeFormalGenericTypes.setAllFalse();
nodeEffectivesTypes.setAllFalse();

nodeClassName.nodeValue = nodeClassName.getParent().getParent().toString();

//-----

/*
 * Here we create the nodeInterfaceName
 */

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent nodeInterfaceName = new NodeComponent(nameText, null,
parametersComponentLanguageNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
parametersInterfaceNode.add(nodeInterfaceName, treeModel);

/**
 * We set the character of the node
 */
/**
 * @see NodeComponent#setAllFalse()
 */
nodeInterfaceName.setAllFalse();

//-----

// The parameters in parametersExtendsNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent parametersExtendsNode = new NodeComponent(parametersText,
extendsNode.getType());

/**
```



```
* @see NodeComponent#add(NodeComponent, DefaultTreeModel)
*/
extendsNode.add(parametersExtendsNode, treeModel);

/*
 * We set the character of the node
 */
parametersExtendsNode.setAddable(value);
parametersExtendsNode.setRemovable(false);
parametersExtendsNode.setCopiable(false);
parametersExtendsNode.setModifiable(false);

//-----

/*
 * Here we create the nodeExtendsName
 */

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent nodeExtendsName = new NodeComponent(nameText, null,
parametersComponentLanguageNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
parametersExtendsNode.add(nodeExtendsName, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
nodeExtendsName.setAllFalse();

//-----

// The parameters in parametersImplementNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
NodeComponent parametersImplementNode = new NodeComponent(parametersText,
implementNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
implementNode.add(parametersImplementNode, treeModel);

/*
 * We set the character of the node
 */
parametersImplementNode.setAddable(value);
parametersImplementNode.setRemovable(false);
parametersImplementNode.setCopiable(false);
parametersImplementNode.setModifiable(false);
```

```
//-----  
  
/*  
    * Here we Creat the nodeImplementName  
    */  
  
/**  
    * @see NodeComponent#NodeComponent(String, String)  
    */  
    NodeComponent nodeImplementName = new NodeComponent(nameText, null,  
parametersComponentLanguageNode.getType());  
  
/**  
    * @see NodeComponent#add(NodeComponent, DefaultTreeModel)  
    */  
    parametersImplementNode.add(nodeImplementName, treeModel);  
  
/**  
    * We set the caracter of the node  
    *  
    * @see NodeComponent#setAllFalse()  
    */  
    nodeImplementName.setAllFalse();  
  
//-----  
  
// The parameters in parametersAgregatesNode  
  
/**  
    * @see NodeComponent#NodeComponent(String, String)  
    */  
    NodeComponent parametersAgregatesNode = new NodeComponent(parametersText,  
agregatesNode.getType());  
  
/**  
    * @see NodeComponent#add(NodeComponent, DefaultTreeModel)  
    */  
    agregatesNode.add(parametersAgregatesNode, treeModel);  
  
/*  
    * We set the caracter of the node  
    */  
    parametersAgregatesNode.setAddable(value);  
    parametersAgregatesNode.setRemovable(false);  
    parametersAgregatesNode.setCopiable(false);  
    parametersAgregatesNode.setModifiable(false);  
  
//-----  
  
/*  
    * Here we Creat the nodeAgregatesName  
    */  
  
/**  
    * @see NodeComponent#NodeComponent(String, String)  
    */  
    NodeComponent nodeAgregatesName = new NodeComponent(nameText, null,  
parametersComponentLanguageNode.getType());
```

```
/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
parametersAgregatesNode.add(nodeAgregatesName, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
    nodeAgregatesName.setAllFalse();

    //-----

// The parameters in parametersComposesNode

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
    NodeComponent parametersComposesNode = new NodeComponent(parametersText,
composesNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
composesNode.add(parametersComposesNode, treeModel);

/*
 * We set the character of the node
 */
parametersComposesNode.setAddable(value);
parametersComposesNode.setRemovable(false);
parametersComposesNode.setCopiable(false);
parametersComposesNode.setModifiable(false);

//-----

/*
 * Here we Create the nodeComposesName
 */

/**
 * @see NodeComponent#NodeComponent(String, String)
 */
    NodeComponent nodeComposesName = new NodeComponent(nameText, null,
parametersComponentLanguageNode.getType());

/**
 * @see NodeComponent#add(NodeComponent, DefaultTreeModel)
 */
parametersComposesNode.add(nodeComposesName, treeModel);

/**
 * We set the character of the node
 *
 * @see NodeComponent#setAllFalse()
 */
    nodeComposesName.setAllFalse();
}
}
```

## Annexe 8 : La classe PopupListener

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

/**
 *
 * This class extends fromm the MouseAdapter class
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public class PopupListener extends MouseAdapter implements ConstantesString
{
    /**
     * This are the local variable
     */
    private NodeComponent nodeDrag;
    private NodeComponent nodeDrop;
    private NodeComponent referNode;
    private JTree languageTree;
    private DefaultTreeModel languageModel;
    private Object nodeInfo;
    private RightPanel theRightPanel;

    /**
     * Class constructor : it creat a new popup Listener
     *
     * @post Creat a new popup listener on the current tree
     * @pre refTree is a JTree attribut, treeModel is a
     *         DefaultTreeModel attribut, jRightPanel is a RightPanel attribut
     *
     * @param refTree the curent tree
     * @param treeModel the model of the current tree
     * @param jRightPanel The panel where the informations are show to the user
     * @since JDK1
     */
    public PopupListener(JTree refTree, DefaultTreeModel treeModel, RightPanel jRightPanel)
    {
        super();
        languageTree = refTree;
        languageModel = treeModel;
        theRightPanel = jRightPanel;
    }

    /**
     * @param MouseEvent the curent tree
     * @since JDK1
     */
}
```

```
*/
public void mousePressed(MouseEvent e)
{
    /**
     * call maybeShowPopup method in this class
     *
     * @see PopuoListener#maybeShowPopup(MouseEvent)
     */
    maybeShowPopup(e);

    /*
     * "Fonction pour l'affichage sur le panneau droite"
     */
    TreePath currentSelection = languageTree.getSelectionPath();

    //System.out.println(currentSelection);

    if(currentSelection != null)// if no node are select
    {
        referNode = (NodeComponent)(currentSelection.getLastPathComponent());

        if(referNode.children().hasMoreElements() == false)
        {
            /**
             * @see RightPanel#removeAll()
             * @see RightPanel#addFile(JTree)
             */
            theRightPanel.removeAll();
            /*ComponentDescription cDesRef=referNode.cDes;*/
            theRightPanel.addFile(languageTree/*, cDesRef*/);
        }
    }
    else
    {
        if(referNode.getType()!=null && ((NodeComponent)referNode.getParent()).getType()==null)
        {
            /**
             * @see RightPanel#removeAll()
             * @see RightPanel#addSubData(JTree)
             */
            theRightPanel.removeAll();
            theRightPanel.addSubData(languageTree);
        }
        else
        {
            /**
             * @see RightPanel#removeAll()
             * @see RightPanel#addDir(JTree)
             */
            theRightPanel.removeAll();
            theRightPanel.addDir(languageTree);
        }
    }
}

/**
 * @param MouseEvent
 * @since JDK 1
 */
public void mouseReleased(MouseEvent e)
```

```
    {
        // call maybeShowPopup method in this class
        maybeShowPopup(e);
    }

    /**
     * @param MouseEvent
     * @since JDK 1
     */
    public void maybeShowPopup(MouseEvent e)
    {
        if(e.isPopupTrigger())
        {
            TreePath currentSelection = languageTree.getSelectionPath();

            //System.out.println(currentSelection);

            if(currentSelection != null)
            {
                referNode = (NodeComponent)(currentSelection.getLastPathComponent());
                MenuPopup mP = new MenuPopup(referNode, languageModel,
languageTree);

                mP.show(e.getComponent(), e.getX(), e.getY());
                mP.setVisible(true);
            }
        }
    }
}
```

## Annexe 9 : La classe ConstantesString

```
/**
 * This class save all the constante
 * of text use on the OFL MEta software
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public interface ConstantesString
{
    /*
     * The string where are save
     * the name of OFL Meta menu
     */
    public String fileText= "File";
    public String optionText= "Option";
    public String helpText= "Help";

    /*
     * The string where are save
     * ShortCut to the menu
     */
    public char fileShortCut= 'F';
    public char optionShortCut= 'O';
    public char aboutShortCut= 'A';

    /*
     * The string where are save
     * the command use in the "File" Menu
     */
    public String newText= "New";
    public String openText= "Open";
    public String importText= "Import";
    public String saveAsText= "Save As";
    public String saveText= "Save";
    public String quitText= "Exit";

    /*
     * The string where are save
     * the command use in the "Option" Menu
     */
    public String addText= "Add";
    public String removeText= "Remove";
    public String renameText= "Rename";
    public String copyText= "Copy";
    public String pasteText= "Paste";
    public String collapseText= "Collapse";
    public String expandText= "Expand";

    /*
     * The string where are save
     * the command use in the "Help" Menu
     */
    public String aProposText= "A Propos";
}
```

```
public String helpContentsText= "Help contents";

/*
 * The string where are save
 * the name of "aide qui apparait "
 */
public String newTTPText= "New Language";
public String openTTPText= "Open Language";
public String saveAsTTPText= "Save Language As";
public String saveTTPText= "Save Language";
public String addTTPText= "Add Node";
public String removeTTPText= "Remove Node";

/*
 * The string where are save
 * the name of the OFL Meta Library language name
 */
public String nameLibrary1="C++";
public String nameLibrary2="Java";

/*
 * The string where are save
 * the name of our icons
 */
public String newIcon= "New.gif";
public String openIcon= "Open.gif";
public String saveAsIcon= "Save As.gif";
public String saveIcon= "Save.gif";
public String addIcon= "Add.gif";
public String removeIcon= "Remove.gif";
public String helpIcon= "Help.gif";
public String SoftIcon= "Help.gif";

/*
 * The string where are save
 * the name node using when we creat the language
 */
public String ComponentLanguageText="Language-Component";
public String ComponentDescriptionText="Descriptions-Components";
public String ComponentRelationText="Relationships-Components";
//-----
public String parametersText = "Parameters";
public String treeNodeText = "Node";
public String nameText = "Name";
public String classeText = "Classe";
public String interfaceText = "Interface";
// public String importText = "Import"; --> already creat
public String useText = "Use";
public String extendsText = "Extends";
public String implementText = "Implement";
public String agregatesText = "Agregates";
public String composesText = "Composes";
public String featuresText = "Features";
public String genericText = "Generic";
public String formalGenericTypesText = "Formal Generic Types";
public String effectivesTypesText = "Effectives Types";
//-----
public String lcText = "LC";
public String dcText = "DC";
public String rcText = "RC";
```



```
/*  
 * Diver  
 */  
public String exitMsgBoxText= "A bientôt sur OFL-META";  
public String statusBarText="For Help Press F1";  
  
public String defaultTreeText="OFL Meta";  
public String treeLanguage="Language";  
public String treeLibrary="Libraries";  
}
```

## Annexe 10 : La classe OFL-Meta

```
import javax.swing.UIManager;
import java.awt.*;
import javax.swing.*;

/**
 * Test the OFL-Meta application
 *
 * @author Nicolas Coulomb
 * @author Baptiste Gasiglia
 * @version OFLMeta version 1.0
 * @since JDK1
 */
public class Test implements ConstantesString
{
    public static void main(String args[])
    {
        try
        {
            // standard look-and-feel
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        Fenetre mainFrame = new Fenetre();
    }
}
```