

Arbres

- Un premier exemple
- Contenu
- Rendu
- Ecouteurs
- Parcours
- Un deuxième exemple

JTree

- Description hiérarchique de données
- Sept autres classes utilisées
 - **TreeModel** : contient les données figurant dans l'arbre
 - **TreeNode** : implémentation des noeuds et de la structure **d'arbre**
 - **TreeSelectionModel** : contient le ou les noeuds sélectionnés
 - **TreePath** : un tel objet contient un chemin (de la racine vers le sommet sélectionné par exemple)
 - **TreeCellRenderer** : est appelé pour dessiner un noeud
 - **TreeCellEditor** : l'éditeur pour un noeud est éditable
 - **TreeUI** : *look-and-feel*

JTree

- Un arbre est créé à partir d'un **TreeModel**
- Il existe plusieurs modèles de sélection
 - sélection d'un seul élément
 - sélection de plusieurs éléments contigus
 - sélection de plusieurs éléments disparates
- On peut indiquer un **CellRenderer** pour afficher une cellule de façon particulière.
- On peut indiquer un **CellEditor** pour changer la valeur d'une cellule

```
interface TreeModel {  
    ...  
    public Object getChild(Object parent, int index);  
    public Object getRoot();  
    public boolean isLeaf(Object node);  
    ...  
}
```

Arbres



Arbre1.bat

- **JTree** fournit une vue du modèle
- Le modèle d'arbre est en deux étapes:

```
interface TreeModel
class DefaultTreeModel implements TreeModel
```

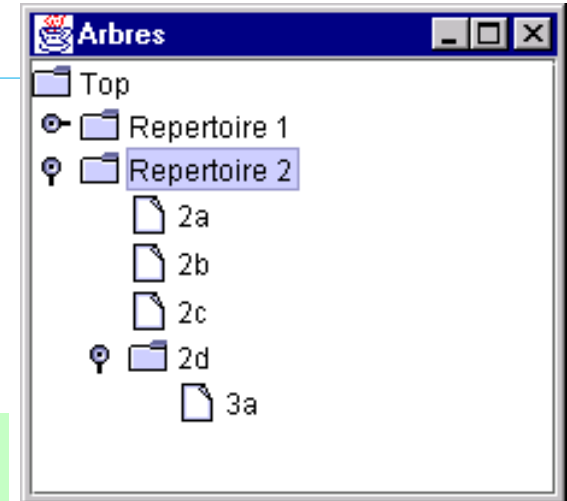
- Le modèle des noeuds est en trois étages:

```
interface TreeNode
interface MutableTreeNode extends TreeNode
class DefaultMutableTreeNode implements MutableTreeNode
```

- Constructeurs

- une feuille
 - ◆ peut recevoir des fils ?
 - ◆ reste sans fils ?

```
JTree()
JTree(TreeNode racine)
JTree(TreeNode racine, boolean enfantsPermis)
JTree(TreeModel modele)
JTree(TreeModel modele, boolean enfantsPermis)
```



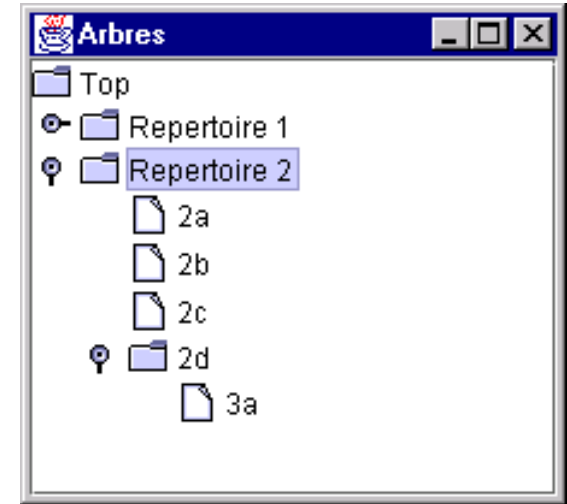
Exemple

```

class Arbre extends JPanel {
    JTree tree;
    public Arbre() {
        DefaultMutableTreeNode top, noeud, fils, n;
        top = new DefaultMutableTreeNode("Top");
        tree = new JTree(top);

        noeud = new DefaultMutableTreeNode("Repertoire 1");
        top.add(noeud);
        n = new DefaultMutableTreeNode("1a"); noeud.add(n);
        n = new DefaultMutableTreeNode("1b"); noeud.add(n);
        ...
        noeud = new DefaultMutableTreeNode("Repertoire 2");
        top.add(noeud);
        n = new DefaultMutableTreeNode("2a"); noeud.add(n);
        ....
        fils = new DefaultMutableTreeNode("2d"); noeud.add(fils);
        n = new DefaultMutableTreeNode("3a"); fils.add(n);
    }
    ...
}

```



Contenu

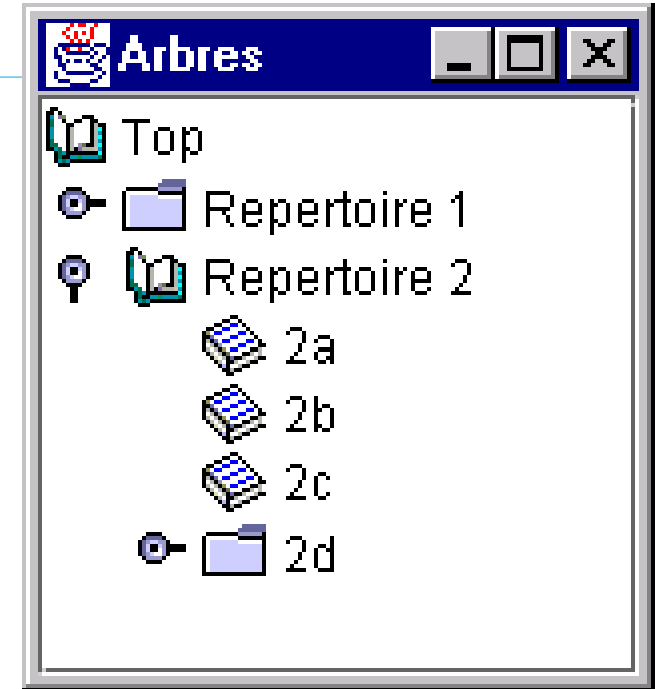
- Le contenu d'un noeud est appelé *user object*
- C'est un objet
- A l'affichage, la méthode `toString()` d'un noeud délègue à la méthode `toString()` du contenu.

Rendu

- Un **DefaultTreeCellRenderer** s'occupe du rendu. Il peut être modifié par
 - des fonctions utilitaires
 - par une redéfinition

```
DefaultTreeCellRenderer rendu ;
```

```
rendu = (DefaultTreeCellRenderer) tree.getCellRenderer();  
rendu.setOpenIcon(new ImageIcon("Opened.gif"));  
rendu.setLeafIcon(new ImageIcon("Leaf.gif"));
```



Sélection

- Un **TreeSelectionListener** rapporte tous les changements dans les sélections
- De nombreuses fonctions utilitaires



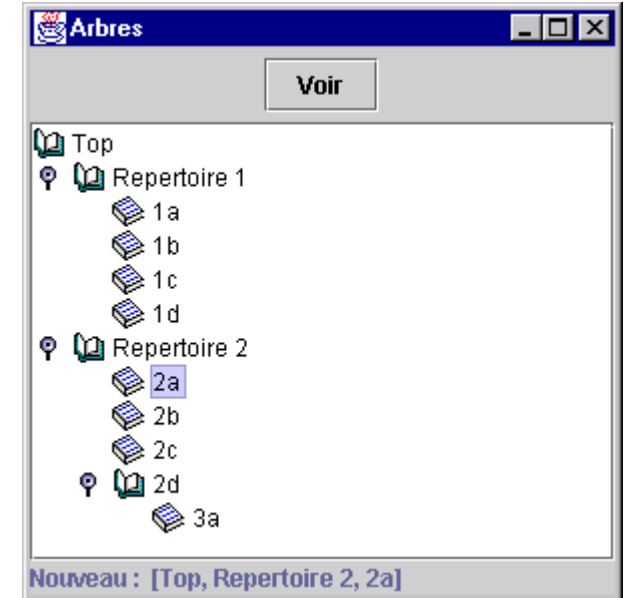
```
tree.addTreeSelectionListener(new Selecteur());

class Selecteur implements TreeSelectionListener {
    public void valueChanged( TreeSelectionEvent e ) {
        message.setText( "Nouveau : " + e.getNewLeadSelectionPath() );
    }
}
```

Parcours

- On parcourt un arbre par une énumération
- Il en existe trois
 - **breadthFirstEnumeration**
 - **depthFirstEnumeration**
 - **postorderEnumeration**
 - **preorderEnumeration**

```
public void actionPerformed(ActionEvent ev) {
    DefaultMutableTreeNode n, top;
    Enumeration e;
    top = (DefaultMutableTreeNode)tree.getModel().getRoot();
    System.out.println("\n En largeur");
    e = top.breadthFirstEnumeration();
    while (e.hasMoreElements()) {
        n = (DefaultMutableTreeNode) e.nextElement();
        System.out.println(n.getUserObject()+" ");
    }
}
```

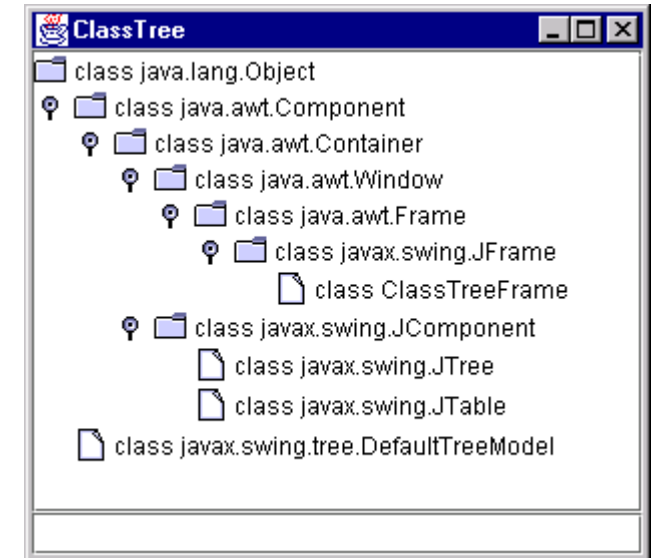


Exemple : un arbre de classes



ClassTree.bat

- Dans cette application, on entre un nom de classe dans la zone de texte, et la classe s'insère dans la hiérarchie des classes.
- La classe **Class** permet de connaître la classe mère.
- On n'insère une classe que si elle n'est pas déjà dans l'arbre.



Constructeur de l'arbre

```
class ClassTreeFrame extends JFrame implements ActionListener {
    private DefaultMutableTreeNode root;
    private DefaultTreeModel model;
    private JTree tree;
    private JTextField textField;

    public ClassTreeFrame() {
        setTitle("ClassTree");
        root = new DefaultMutableTreeNode(Object.class);
        model = new DefaultTreeModel(root);
        tree = new JTree(model);

        addClass(getClass());
        getContentPane().add(new JScrollPane(tree), "Center");
        textField = new JTextField();
        textField.addActionListener(this);
        getContentPane().add(textField, "South");
    }
    ...
}
```

- C'est `addClass(Class c)` qui fait l'insertion

Ajouter une classe

```
public DefaultMutableTreeNode addClass(Class c) {  
  
    if (c.isInterface() || c.isPrimitive()) return null; pas les interfaces  
  
    findUserObject(c) cherche c dans tree  
    DefaultMutableTreeNode node = findUserObject(c);  
    if (node != null)  
        return node;  
  
    Class s = c.getSuperclass(); classe mère  
  
    DefaultMutableTreeNode parent = addClass(s); appel récursif  
  
    DefaultMutableTreeNode newNode = new DefaultMutableTreeNode(c);  
    model.insertNodeInto(newNode, parent, parent.getChildCount()); à la fin  
  
    développe l'arbre pour que le noeud soit visible  
    TreePath path = new TreePath(model.getPathToRoot(newNode));  
    tree.makeVisible(path);  
  
    return newNode;  
}
```

Trouver un noeud dans un arbre

- Un simple parcours, en largeur par exemple

```
public DefaultMutableTreeNode findUserObject(Object obj) {  
    Enumeration e = root.breadthFirstEnumeration();  
    while (e.hasMoreElements()) {  
        DefaultMutableTreeNode node = (DefaultMutableTreeNode) e.nextElement();  
        if (node.getUserObject().equals(obj))  
            return node;  
    }  
    return null;  
}
```

Lire le nom de la classe

- On fait confiance à Java...

```
public void actionPerformed(ActionEvent event) {
    String text = textField.getText();
    try {
        Class c = Class.forName(text); essayons
        addClass(c);
        textField.setText("");
    }
    catch (ClassNotFoundException e) {
        Toolkit.getDefaultToolkit().beep(); si la classe n'existe pas
    }
}
```