6

(Styled Text Components)

1 .

1 .

AWT 1

. .

. ,

.

6-1

. .

. , ,

.

6-1:

6-2 : , ,

.

JTree . JTree (JTree helper

) . JTree

.

.

Kim Topley Core Java Foundation classes[Prentice-Hall 1998] David M.Geary

Graphic Java 2 .

.(6-3   )   .

.

.   .

.

6-3 :

.(323

6-5)   JTree   / /

.

. JTree   .

TreeModel model = . . . ;

JTree tree = new JTree(model);

:   .

JTree(Object[] nodes)

JTree(Vector nodes)

JTree(Hashtable nodes) //   .

.   .

.

?   TreeModel

.

DefaultTreeModel   .

Default tree model   .

TreeNode root = . . .;

DefaultTreeModel model = new DefaultTreeModel (root);

TreeNode   .

default tree model   .

DefaultMutableTreeNode   .   TreeNode

MutatleTreeNode   .(6-4)

6-4 :

default mutable tree　　　　　　　　　　.

　　.　　　　　　　　　　　　　　　　　　　toString

　　　.

　　　　　　　　　　　　　　　　　　.

　　　　　　　　　　　　　　.　　　　,

　　　　　　File　　　　　　　　　　　.

　　　　　　　　　　　　　setUserObject

　　　.

　　DefaultMutableTreeNode node

　　　　　= new DefaultMutableTreeNode("Texas");

　　node.setUserObject("California");

　　　　　　　　/　　　　.　　　　　　　add

　　　　　　　.

DefaultMutableTreenode root

　　　= new DefaultMutableTreeNode("World");

DefaultMutableTreeNode country

　　　= new DefaultMutableTreeNode("USA");

root.add(country);

DefaultMutableTreeNode state

　　　= new DefaultMutableTreeNode("Callfornia");

country.add(state);

　　　　　　　　6-5　　　　.

6-5 :

　　　　　　　　　.　　　　　　DefaultTreeModel

.　　　，　　　JTree　　　.

　　DefaultTreeModel treeModel = new DefaultTreeModel(root);

　　JTree tree = new JTree(treeModel);

　　　　JTree　　　　　　.

　　　　default tree model　　　.

　　6-1　　　　　.

6-1 : SimpleTree.java

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


public class SimpleTree
{  public static void main(String[] args)
   {   JFrame frame = new SimpleTreeFrame();
       frame.show();
   }
}


class SimpleTreeFrame extends JFrame
{  public SimpleTreeFrame()
   {   setTitle("SimpleTree");
      setSize(300, 200);
      addWindowListener(new WindowAdapter()
         {  public void windowClosing(WindowEvent e)
            {   System.exit(0);
            }
         } );

   // set up tree model data

      DefaultMutableTreeNode root
         = new DefaultMutableTreeNode("World");
      DefaultMutableTreeNode country
         = new DefaultMutableTreeNode("USA");
       root.add(country);
      DefaultMutableTreeNode state
         = new DefaultMutableTreeNode("California");
       country.add(state);
      DefaultMutableTreeNode city
         = new DefaultMutableTreeNode("San Jose");
       state.add(city);
```

```
        city = new DefaultMutableTreeNode("Cupertino");
         state.add(city);
        state = new DefaultMutableTreeNode("Michigan");
         country.add(state);
        city = new DefaultMutableTreeNode("Ann Arbor");
         state.add(city);
        country = new DefaultMutableTreeNode("Germany");
         root.add(country);
        state = new DefaultMutableTreeNode("Schleswig-Holstein");
         country.add(state);
        city = new DefaultMutableTreeNode("Kiel");
         state.add(city);


     // construct tree and put it in a scroll pane

     JTree tree = new JTree(root);
     Container contentPane = getContentPane();
      contentPane.add(new JScrollPane(tree));
   }
}
```

6-6   .

.   (   )   .

          .(6-7)
                              .   .

     :   ,   .
     .(a.k.a. "Metal")   (”+”
     "-")   .(6-8   )

6-6 :

6-7 :

6-8 :

.

.

Tree.putClientProperty("JTree.lineStyle", "Angled");

6-9   .

6-9 :

6-10   .

.

.

6-10 :

.

.

Tree.setShowsRootHandles(true);

6-1   .   .

6-11:

.

.   ,   .

Tree.setRootVisible(false);

6-12   . "USA"   "Germany"   .

.

6-12 :   (forest)

.

.

6-13 :

.　　　　　　　　　　　,

.

.

.

.　　　　　　　isLeaf　　　　　true　　　　　　　　　.

.

DefaultMutableTreeNode　　　　　isLeaf　　　　　　　　　　　　　　　　true

.

.

.　　　　' '　　　　　"Montana"

.　　　　　　　　　　　　　　　　　.

.

JTree　　　　　　　　　　.　　　　　　　　　　.

.　　　　　　　　.

.

node.setAllowsChildren(false)

,　　　　　　　　　　　　　　　　　　" 　　　　　"

. DefaultTreeModel　　　　setAsksAllowsChildren　　　.

model.setAsksAllowsChildren(true);

.

" 　　　　"

.

JTree tree = new JTree(root, true);

//　　　　　　　　　　　　　　　.

javax.swing.Jtree

- JTree(TreeModel model)

.

- JTree(TreeNode root)
- JTree(TreeNode root , Boolean askAllowChildren)

　　　　　　　　　　　　　　　　　　　　　　　　　　　　　.

　　　　　　　　　:　　　　　root

　　　　　　　　　　　　　askAllowChildren

　　　　　　　　　　　　　　　　　　　　　　　　true

　　　　　　　　　　　　　　　　　　　.

- void setShowsRootHandles(Boolean b)

　　　　b　　true　　　,

　　　　　　　.

- void setRootVisible(boolean b)

　　　　b　　true　　　,　　　　　　　　　　.　　　　　　　　　　.

javax.swing.tree.TreeNode

- boolean isLead()

　　　　　　　　　　　　　　　　　true　　　　　.

- boolean getAllowChildren()

　　　　　　　　　　　　　　true　　　　.

javax.swing.tree.MutableTreeNode

- void setUserObject(Object userObject)

　　　　　　　　　　　　　　　　.

javax.swing.tree.TreeModel

- boolean isLeaf(TreeNode node)

　　　node　　　　　　　　　　　true　　　　.

javax.swing.tree.DefaultTreeModel

- void setAsksAllowsChildren(boolean b)

　　　b　　true　　,　　　　getAllowsChildren　　　flase

　　　　　.　　　　　　　isLeaf　　　true

　　　　　.

javax.swing.tree.DefaultMutableTreeNode

- DefaultMutableTreeNode(Object userObject)

　　　　　　　　　　　　　.

- void add(MutableTreeNode child)

　　　　　　　　　　.

- void setAllowsChildren(boolean b)

　　　b　　true　　,　　　　　　　　　.

javax.swing.tree.JComponent

- void putClientProperty(object key, Object value)

/   .

.

.   6-14

.   "Add Sibling" or "Add Child"

.   "Delete"   .

6-14 :

. JTree

.

.

.(6-15)

6-15:

JTree   . TreeNode   getParent

?   JTree   TreeNode

.   TreeModel   .

DefaultTreeModel   .   TreeNode

.

getParent   getChild   .

.   . JTree

.   JTree

.

TreePath   (TreeNode   )   .   JTree

TreePath   .

getLastPastComponent   .

JTree   getSelectionPath   . TreePath

.

    TreePath selectionPath = tree.getSelectionPath();

    DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode)

SelectionPath.getLastPathComponent();

,

.

DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode)
Tree.getlastSelectedPathComponent();

getSelectedNode   .   .

:   JTree

.   JTree   .
(0   )   .

.   .
JTree   .

,   .

.

selectedNode.add(newNode);   // NO!

.   .

DefaultTreeModel   insertNodeInto

.

.

model.insertNodeInto(newNode, selectedNode,
selectedNode.getChildCount());
removeNodeFromParent   .
model.removeNodeFromParent(selectedNode);

.

model.nodeChanged(changedNode);
DefaultTreeModel   .

.(Kim Topley
Core Java Foundation classes   .)

: DefaultTreeModel   reload   .
reload   .

.

.

.

. ,

.

.   JTree   makeVisible   . MakeVisible

.

.

DefaultTreeModel   getPathToRoot   .

TreeNode[]   .

TreePath   .

,   .

TreeNode[] nodes = model.getPathToRoot(newNode);

TreePath path = new TreePath(nodes);

tree.makeVisible(path);

: DefaultTreeModel   JTree   TreePath

. JTree

.

.

.

makeVisible

tree.scrollPathToVisible(path);

.

.(6-16)

6-16 :

.   ,

tree.setEditable(true);

.

DefaultCellEditor   default cell editor   .

.

.

6-2   .

.

.

6-2 : TreeEditTest.java

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

public class TreeEditTest
{   public  static void main(String[] args)
    {   JFrame  frame = new TreeEditFrame();
         frame.show();
    }
}

class TreeEditFrame extends JFrame
    implements  ActionListener
{   public TreeEditFrame()
    {    setTitle("TreeEditTest");
       setSize(300,  200);
       addWindowListener(new WindowAdapter()
          {   public  void  windowClosing(WindowEvent e)
              {    System.exit(0);
              }
          } );

       // construct  tree

       TreeNode root = makeSampleTree();
       model = new DefaultTreeModel(root);
```

```java
      tree = new JTree(model);
       tree.setEditable(true);

   // add scroll pane with tree to content pane

     Container contentPane = getContentPane();
     JScrollPane scrollPane = new JScrollPane(tree);
      contentPane.add(scrollPane, "Center");

     // make button panel

     JPanel panel = new JPanel();
     addSiblingButton = new JButton("Add Sibling");
      addSiblingButton.addActionListener(this);
      panel.add(addSiblingButton);
     addChildButton = new JButton("Add Child");
      addChildButton.addActionListener(this);
      panel.add(addChildButton);
     deleteButton = new JButton("Delete");
      deleteButton.addActionListener(this);
      panel.add(deleteButton);
      contentPane.add(panel, "South");
   }

public TreeNode makeSampleTree()
{   DefaultMutableTreeNode root
       = new DefaultMutableTreeNode("World");
    DefaultMutableTreeNode country
       = new DefaultMutableTreeNode("USA");
     root.add(country);
    DefaultMutableTreeNode state
        = new DefaultMutableTreeNode("California");
     country.add(state);
    DefaultMutableTreeNode city
       = new DefaultMutableTreeNode("San Jose");
     state.add(city);
```

```java
      city = new DefaultMutableTreeNode("Cupertino");
       state.add(city);
      state = new DefaultMutableTreeNode("Michigan");
       country.add(state);
     city = new DefaultMutableTreeNode("Ann Arbor");
       state.add(city);
     country = new DefaultMutableTreeNode("Germany");
       root.add(country);
     state = new DefaultMutableTreeNode("Schleswig-Holstein");
       country.add(state);
     city = new DefaultMutableTreeNode("Kiel");
       state.add(city);
       return root;
   }

public void actionPerformed(ActionEvent event)
{   DefaultMutableTreeNode selectedNode
       = (DefaultMutableTreeNode)
           tree.getLastSelectedPathComponent();

   if (selectedNode == null) return;

    if (event.getSource().equals(deleteButton))
    {   if (selectedNode.getParent() != null)
           model.removeNodeFromParent(selectedNode);
        return;
    }

  // add new node as sibling or child

    DefaultMutableTreeNode newNode
       = new DefaultMutableTreeNode("New");

    if (event.getSource().equals(addSiblingButton))
    {   DefaultMutableTreeNode parent
           = (DefaultMutableTreeNode)selectedNode.getParent();
```

```
         if (parent != null)
          {   int selectedIndex = parent.getIndex(selectedNode);
               model.insertNodeInto(newNode, parent,
                   selectedIndex + 1);
             }
       }
      else if (event.getSource().equals(addChildButton))
       {   model.insertNodeInto(newNode, selectedNode,
               selectedNode.getChildCount());
         }

      // now display new node

      TreeNode[] nodes = model.getPathToRoot(newNode);
      TreePath path = new TreePath(nodes);
        tree.scrollPathToVisible(path);
    }

   private DefaultTreeModel model;
   private JTree tree;
   private JButton addSiblingButton;
   private JButton addChildButton;
   private JButton deleteButton;
   private JButton editButton;
}
```

javax.swing.JTree

- TreePath getSelectionPath()

                                                   (
).   null   .
- Object getLastSelectedPathComponent()

                                              (
).   null   .
- void makeVisible(TreePath path)

                                        .

- void scrollPathToVisible(TreePath path)

.

.

javax.swing.tree.TreePath

- object getLastPathComponent()

. ,

javax.swing.tree.TreeNode

- TreeNode getParent()

.

- TreeNode getChildrenAt(int index)

.   0   getChildCount() – 1

.

- int getChildrenCount()

.

- Enumeration children()

.

javax.swing.tree.DefaultTreeNode

- void insertNodeInto(MutableTreeNode newChild, MutableTreeNode parent, int index)

parent .

- void removeNodeFromParent(MutableTreeNode node)

node .

- void nodeChanged(TreeNode node)

node .

- void nodesChanged(TreeNode parent, int[] changedChildIndexes)

parent

.

- void reload()

.

.

. DefaultMutableTreeNode

.

breadthFirstEnumeration   depthFirstEnumeration

nextElement

.   6-17   .
                          .   .
              .   .
                                                                    .   ,
                                        .   .
                          .

                          (postorder)   . postOrderTraversal

depthFirstTraversal   .

                                                    PreOrderTraversal   .

                                    .

        Enumeration breadthFirst = node.breadthFirstEnumeration();

        While (breadthFirst.hasMoreElements())

          do something with breadthFirst.nextElement();

6-17 :

    ,

    pathFromAncestorEnumeration   .

    getParent   .

                                                                    .

                  .   .

                                        .(6-18)

                                                                                    .

                                            Class   .

                                    .

                  .

                        .

          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;

           }
```

6-18 :

.   .

.

                         tree cell renderer   .   JTree

DefaultTreeCellRenderer   . DefaultTreeCellRenderer

    JLabel   .   .

     :   " "   .

                                                              .

   1.   DefaultTreeCellRenderer   ,   .

                                                     .

   2.   DefaultTreeCellRenderer   ,   ,

                                       .

   3.   TreeCellRenderer

              .

                                                       .   DeaultTreeCellRenderer

                                                    .

     DefaultTreeCellRenderer renderer
              = new DefaultTreeCellRenderer();

     renderer.setLeafIcon(new ImageIcon("blue-ball.gif"));

              //

     renderer.setClosedIcon(new ImageIcon("red-ball.gif"));

              //

     renderer.setOpenIcon(new ImageIcon("yellow-ball.gif"));

              //

tree.setCellRenderer(renderer);

338   6-18   .   "ball"

.

.

.

,

.   6-18

.

.

1   9   . TreeCellRenderer

Component getTreeCellRendererComponent(Jtree tree, Object value,

Boolean selected, Boolean expanded,

Boolean leaf, int row, Boolean hasFocus)

.

.   paint

. Paint   Graphics   .

Graphics   .

: paint

.   .

.   ,

JLabel

.

DefaultTreeCellRenderer   getTreeCellRendererComponent   this

. (DefaultTreeCellRenderer   Jlabel   .)

DefaultTreeCellRenderer   .

getTreeCellRendererComponent   .   ,

.   this   .

```
class MyTreeCellRenderer extends DefaultTreeCellRenderer
{   public Component getTreeCellRendererComponent(Jtree tree,
        Object value, boolean selected, Boolean expanded,
        boolean leaf, int row, Boolean hasFocus)
    { super.getTreeCellRendererComponent(tree, value,
```

```
                    selected, expanded, leaf, row, hasFocus) ;
                    DefaultMutableTreeNode node
                        = (DefaultMutableTreeNode)value;
                    look at node.getUserObject();
                    Font font = appropriate font;
                    setFont(font);
                    return this;
                }
            };
```

: getTreeCellRendererComponent   value

.   DefaultMutableTreeNode   .   JTree
                                        .   DefaultMutableTreeNode

.

: DefaultTreeCellRenderer
                        .
                                    .
        .   6-3   .

                                                                                            . 1   9
                                    .   .

                                    .   6-3
            .   .
                                        .
                            .

        ENTER   .((stellar)   ,
            ENTER   .)
                    java.util.Vector   .
                                                                                                        .
            addClass   . (.
                                                                                            .

                        .)

findUserObject

.

.

.

ClassNameTreeCellRenderer　　Class　　　ABSTRACT

.

.

.　　　　　　　　　　　　　　JLabel　　　　　　　　　　　　　　　　.

getTreeCellRendererComponent　　　　　　　　　　　　　　　　　　.

ClassTreeFrame

.

6-3 : ClassTree.java

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

public class ClassTree
{  public static void main(String[] args)
   {   JFrame frame = new ClassTreeFrame();
       frame.show();
   }
}

class ClassTreeFrame extends JFrame
   implements  ActionListener
{  public ClassTreeFrame()
   {   setTitle("ClassTree");
       setSize(300,  200);
       addWindowListener(new  WindowAdapter()
```

```
                     {   public void windowClosing(WindowEvent e)
                         {    System.exit(0);
                         }
                     } );


      // the root of the class tree is Object
      root = new DefaultMutableTreeNode(java.lang.Object.class);
      model = new DefaultTreeModel(root);
      tree = new JTree(model);


      // add this class to populate the tree with some data
        addClass(getClass());


      // set up node icons
       ClassNameTreeCellRenderer renderer
          = new ClassNameTreeCellRenderer();
       renderer.setClosedIcon(new ImageIcon("red-ball.gif"));
       renderer.setOpenIcon(new ImageIcon("yellow-ball.gif"));
       renderer.setLeafIcon(new ImageIcon("blue-ball.gif"));
        tree.setCellRenderer(renderer);


      Container contentPane = getContentPane();
      contentPane.add(new JScrollPane(tree), "Center");


      // new class names are typed into this text field
      textField = new JTextField();
       textField.addActionListener(this);
       contentPane.add(textField, "South");
   }

public void actionPerformed(ActionEvent event)
{  // add the class whose name is in the text field
      try
     {    String text = textField.getText();
          addClass(Class.forName(text));
        // clear text field to indicate success
```

```java
         textField.setText("");
      }
    catch (ClassNotFoundException e)
     {   Toolkit.getDefaultToolkit().beep();
      }
 }


public DefaultMutableTreeNode findUserObject(Object obj)
{  // find the node containing a user object
    Enumeration e = root.breadthFirstEnumeration();
    while (e.hasMoreElements())
     {  DefaultMutableTreeNode node
           = (DefaultMutableTreeNode)e.nextElement();
       if (node.getUserObject().equals(obj))
           return node;
     }
    return null;
 }


public DefaultMutableTreeNode addClass(Class c)
{  // add a new class to the tree

     // skip non-class types
   if (c.isInterface() || c.isPrimitive()) return null;

   // if the class is already in the tree, return its node
    DefaultMutableTreeNode node = findUserObject(c);
   if (node != null) return node;

    // class isn't present--first add class parent recursively

   Class s = c.getSuperclass();

    DefaultMutableTreeNode parent;
   if (s == null)
       parent = root;
```

```
        else
            parent = addClass(s);

      // add the class as a child to the parent
      DefaultMutableTreeNode newNode = new DefaultMutableTreeNode(c);
       model.insertNodeInto(newNode, parent, parent.getChildCount());

       // make node visible
      TreePath path = new TreePath(model.getPathToRoot(newNode));
       tree.makeVisible(path);

       return newNode;
   }

   private DefaultMutableTreeNode root;
   private DefaultTreeModel model;
   private JTree tree;
   private JTextField textField;
}

class ClassNameTreeCellRenderer extends DefaultTreeCellRenderer
{   public Component getTreeCellRendererComponent(JTree tree,
      Object value, boolean selected, boolean expanded,
      boolean leaf, int row, boolean hasFocus)
   {   super.getTreeCellRendererComponent(tree, value,
         selected, expanded, leaf, row, hasFocus);
      // get the user object
      DefaultMutableTreeNode node = (DefaultMutableTreeNode)value;
      Class c = (Class)node.getUserObject();

     // the first time, derive italic font from plain font
      if (plainFont == null)
      {   plainFont = getFont();
        /* the tree cell renderer is sometimes called with a
           label that has a null font
         */
```

```
        if (plainFont != null)

            italicFont = plainFont.deriveFont(Font.ITALIC);

      }


    // set font to italic if the class is abstract
    if ((c.getModifiers() & Modifier.ABSTRACT) == 0)
          setFont(plainFont);
      else
          setFont(italicFont);
      return this;
  }


  private Font plainFont = null;
  private Font italicFont = null;
};
```

javax.swing.tree.DefaultMutableTreeNode

- Enumeration breadthFirstEnumeration()
- Enumeration depthFirstEnumeration()
- Enumeration preOrderTraversal()
- Enumeration postOrderTraversal()

                                                                     .

      BreadthFirst   . DepthFirs

                                                                .

      PostOrderTraversal   depthFirstEnumeration   . PreOrderTraversal

                                                             postOrderTraversal

            .

javax.swing.tree.TreeCellRender

- Component getTreeCellRenderComponent(JTree tree, Object value, Boolean selected, Boolean expanded, Boolean leaf, int row, Boolean hasFocus)

                                      paint   .

                :   tree

                           value

                           selected   true   .

                           expanded   true   .

reaf

true   .

row   .

HasFocus   true

.

javax.swing.tree.TreeCellRender

- void setLeafIcon(Icon icon)
- void setOpenIcon(Icon icon)
- void setClosedIcon(Icon icon)

,   .

.

,   .   6-19

,   .

6-19 :

tree selection listener   .

void valueChanged(TreeSelectionEvent event)

TreeSelectionListener   .

.

.

tree.addTreeSelectionListener(listener);

,

.   JTree

TreeSelectionModel   .   SINGLE_TREE_SELECTION,

CONTIGUOUS_TREE_SELECTION,   DISCONTIGUOUS_TREE_SELECTION

.(Discontiguous   .)   ,

.

int mode = TreeSelectionModel.SINGLE_TREE_SELECTION;

tree.getSelectionModel().setSelectionMode(mode);

.

: .

( ) CTRL

.

SHIFT .

---

getSelectionPaths .

TreePath[] selectedPaths = tree.getSelectionPaths();

(

null .) getSelectionPath .

: TreeSelectionEvent TreePath getPaths

. .

6-4 . 6-3

. .

.

ValueChanged ,

.

.

getFieldDescription . .

6-4 : ClassBrowserTree.java

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


public class ClassBrowserTest
{  public static void main(String[] args)
   {   JFrame frame = new ClassBrowserTestFrame();
       frame.show();
   }
```

```
      }

class ClassBrowserTestFrame extends JFrame
    implements ActionListener, TreeSelectionListener
{   public ClassBrowserTestFrame()
    {   setTitle("ClassBrowserTest");
        setSize(300, 200);
        addWindowListener(new WindowAdapter()
            {   public void windowClosing(WindowEvent e)
                {   System.exit(0);
                }
            } );

        // the root of the class tree is Object
        root = new DefaultMutableTreeNode(java.lang.Object.class);
        model = new DefaultTreeModel(root);
        tree = new JTree(model);

        // add this class to populate the tree with some data
        addClass(getClass());

        // set up selection mode
        tree.addTreeSelectionListener(this);
        int mode = TreeSelectionModel.SINGLE_TREE_SELECTION;
        tree.getSelectionModel().setSelectionMode(mode);

        // this text area holds the class description
        textArea = new JTextArea();

        // add tree and text area to the content pane
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 2));
        panel.add(new JScrollPane(tree));
        panel.add(new JScrollPane(textArea));

        Container contentPane = getContentPane();
```

```
         contentPane.add(panel, "Center");


   // new class names are typed into this text
    textField = new JTextField();
     textField.addActionListener(this);
    contentPane.add(textField, "South");
 }


public void actionPerformed(ActionEvent event)
{  // add the class whose name is in the text field
    try
   {   String text = textField.getText();
       addClass(Class.forName(text ));
     // clear text field to indicate success
       textField.setText("");
   }
   catch (ClassNotFoundException e)
   {   Toolkit.getDefaultToolkit().beep();
   }
 }


public void valueChanged(TreeSelectionEvent event)
 {   // the user selected a different node--update description
   TreePath path = tree.getSelectionPath();
   if (path == null) return;
    DefaultMutableTreeNode selectedNode
        = (DefaultMutableTreeNode)path.getLastPathComponent();
   Class c = (Class)selectedNode.getUserObject();
   String description = getFieldDescription(c);
    textArea.setText(description);
 }


public DefaultMutableTreeNode findUserObject(Object obj)
{  // find the node containing a user object
   Enumeration e = root.breadthFirstEnumeration();
   while (e.hasMoreElements())
```

```
    {   DefaultMutableTreeNode node
            = (DefaultMutableTreeNode)e.nextElement();
        if (node.getUserObject().equals(obj))
            return node;
    }
    return null;
}

public DefaultMutableTreeNode addClass(Class c)
{   // add a new class to the tree

    // skip non-class types
    if (c.isInterface() || c.isPrimitive()) return null;

    // if the class is already in the tree, return its node
    DefaultMutableTreeNode node = findUserObject(c);
    if (node != null) return node;

    // class isn't present--first add class parent recursively

    Class s = c.getSuperclass();

    DefaultMutableTreeNode parent;
    if (s == null)
        parent = root;
    else
        parent = addClass(s);


    parent = addClass(s);

    // add the class as a child to the parent
    DefaultMutableTreeNode newNode = new DefaultMutableTreeNode(c);
    model.insertNodeInto(newNode, parent, parent.getChildCount());

    // make node visible
    TreePath path = new TreePath(model.getPathToRoot(newNode));
```

```
        tree.makeVisible(path);


        return newNode;
   }


   public static String getFieldDescription(Class c)
   {  // use reflection to find types and names of fields
      String r = "";
      Field[] fields = c.getDeclaredFields();
     for (int i = 0; i < fields.length; i++)
      {   Field f = fields[i];
         if ((f.getModifiers() & Modifier.STATIC) != 0)
             r += "static ";
         r += f.getType().getName() + " ";
         r += f.getName() + "\n";
       }
      return r;
   }


   private DefaultMutableTreeNode root;
   private DefaultTreeModel model;
   private JTree tree;
   private JTextField textField;
   private JTextArea textArea;
}
```

javax.swing.JTree

- Path getSelectionPath()
- Path[] getSelectionPaths()

.

,   null   .

javax.swing.event.TreeSelectionListener

- void valueChanged(TreeSelectionEvent event)

.

javax.swing.event.TreeSelectionEvent

- TreePath getPath()
- TreePath[] getPaths()

.

,   JTree.getSelectionPaths

.

(6-20   )

.

6-20 :

.   .

.

.   .

DefaultTreeModel   .

.

.

TreeModel   .   JTree

. JTree

.

Object getRoot()

int getChildCount(Object parent)

Object getChild(Object parent, int index)

JTree   TreeModel   .

. TreeModel

JTree   .

TreeModel   getChild   .

int getIndexOfChild(Object parent, Object child)

,   . 6-5   .

　　JTree   .

　　boolean isLeaf(Object node)

　　　　　　　　　　　TreeModelListener   .

　　　　　　　　　　　　　　　　　　　.

　　void addTreeModelListener (TreeModelListener l)

　　void removeTreeModelListener (TreeModelListener l)

　6-5   .

　　　　　　　　　　　　　　TreeModelListener

　　void treeNodesChanged(TreeModelEvent e)

　　void treeNodesInserted(TreeModelEvent e)

　　void treeNodeRemoved(TreeModelEvent e)

　　void treeStructureChanged(TreeModelEvent e)

　4   .

TreeModelEvent   .

　　　　　　　　　　　　　　　　　　　　　　　　.

　　　6-5   .

　　　　　　　.

TIP :   javax.swing.EventListenerList

　　　　　　　　　　　　. API

.

　　　　　　　　　　　　　　　　　　　　　　.

　　void valueForPathChanged(TreePath path, Object newValue)

　　　　　　　　　　　　　　　　.

　　　　　　　　　　　　　　　　　.

　　　　　　　　.

　　Object getRoot()

　　int getChildCount(Object parent)

　　Object getChild(Object parent, int index)

　　　　　　　　　　　.   6-5

.   .   .

　Variable   .

: DefaultTreeModel   Variable

DefaultMutableTreeNode   .

,

Employee joe:

.

Employee.class,   "joe"   joe   .

Variable   .

Variable v = new Variable(Employee.class, "joe",   joe);

(primitive)   .

new Variable(double.class, "salary", new Double(salary));

..   ,

ArrayList   . Class   GetFields

getFields   . Variable

. Variable   getFields   .

, Variable   toString   .

Variable   .   Variable

.

:   .

.   "   "   .

.   .

```
public Object getRoot()
{          return root;
}


public int getChildCount (Object parent)
{          return ((Variable)parent).getFields().size();
}
```

getChild   Variable   .

getType   getName   .

f.get(parentValue)   .

IllegalAcessException   .   Variable

.   .

getChild   .

```
        public object getChild(Object parent, int index)
       { ArrayList fields = ((Variable)parent).getFields();
         Field f = (Field) fields.get(index);
         Object parentValue = ((Variable)parent).getValue();
          try
         { return new Variable(f.getType(), f.getName(),
                 f.get(parentValue));
          }
         catch(IllegalAccessException e)
         { return null;
          }
       }
```

                   JTree   .

         .   6-5   .

    .   . WeakReference

       . referent   .

     .   WeakReference

       .   .

      ,   .

         .

      .

   .

     .

6-5 : ObjectInspectorTest.java

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


public class ObjectInspectorTest
```

```
{   public static void main(String[] args)
   {   JFrame frame = new ObjectInspectorFrame();
       frame.show();
   }
}

class ObjectInspectorFrame extends JFrame
{   public ObjectInspectorFrame()
   {   setTitle("ObjectInspectorTest");
      setSize(300, 200);
      addWindowListener(new WindowAdapter()
         {   public void windowClosing(WindowEvent e)
            {   System.exit(0);
            }
         } );

      // we inspect this frame object

      Variable v = new Variable(getClass(), "this", this);
      ObjectTreeModel model = new ObjectTreeModel();
       model.setRoot(v);

      // construct and show tree

      tree = new JTree(model);
      JScrollPane scrollPane = new JScrollPane(tree);
      Container contentPane = getContentPane();
       contentPane.add(scrollPane, "Center");
   }

   private JTree tree;
}

class ObjectTreeModel implements TreeModel
{   public ObjectTreeModel()
   {   root = null;
```

```
  }

public void setRoot(Variable v)
{   Variable oldRoot = v;
    root = v;
     fireTreeStructureChanged(oldRoot);
 }

public Object getRoot()
 {   return root;
 }

public int getChildCount(Object parent)
 {   return ((Variable)parent).getFields().size();
 }

public Object getChild(Object parent, int index)
 {   ArrayList fields = ((Variable)parent).getFields();
    Field f = (Field)fields.get(index);
    Object parentValue = ((Variable)parent).getValue();
     try
    {   return new Variable(f.getType(), f.getName(),
           f.get(parentValue));
     }
     catch(IllegalAccessException e)
     {   return null;
     }
 }

public int getIndexOfChild(Object parent, Object child)
{   int n = getChildCount(parent);
   for (int i = 0; i < n; i++)
      if (getChild(parent, i).equals(child))
          return i;
    return -1;
 }
```

```java
    public boolean isLeaf(Object node)
    {   return getChildCount(node) == 0;
    }

    public void valueForPathChanged(TreePath path, Object newValue)
    {}

    public void addTreeModelListener(TreeModelListener l)
    {    listenerList.add(TreeModelListener.class, l);
    }

    public void removeTreeModelListener(TreeModelListener l)
    {    listenerList.remove(TreeModelListener.class, l);
    }

    protected void fireTreeStructureChanged(Object oldRoot)
    {    TreeModelEvent event
            = new TreeModelEvent(this, new Object[] {oldRoot});
        Object[] listeners = listenerList.getListenerList();
        for (int i = listeners.length - 2; i >= 0; i -= 2)
            ((TreeModelListener)listeners[i+1]).
                treeStructureChanged(event);
    }

    private Variable root;
    private EventListenerList listenerList
        = new EventListenerList();
}

class Variable
{   public Variable(Class aType, String aName, Object aValue)
    {   type = aType;
        name = aName;
        value = aValue;
        fields = new ArrayList();
```

```
   /* find all fields if we have a class type
      except we don't expand strings and null values
    */

  if (!type.isPrimitive() && !type.isArray() &&
     !type.equals(String.class) && value != null)
   {  // get fields from the class and all superclasses
    for (Class c = value.getClass(); c != null;
         c = c.getSuperclass())
     {   Field[] f = c.getDeclaredFields();
         AccessibleObject.setAccessible(f, true);


        // get all nonstatic fields
      for (int i = 0; i < f.length; i++)
         if ((f[i].getModifiers() & Modifier.STATIC) == 0)
             fields.add(f[i]);
     }
   }
}

public Object getValue()
{   return value;
 }

 public ArrayList getFields()
{   return fields;
 }

public String toString()
{   String r = type + " " + name;
    if (type.isPrimitive())
       r += "=" + value;
    else if (type.equals(String.class))
       r += "=" + value;
    else if (value == null)
```

```
        r += "=null";
      return r;
   }


   private Class type;
   private String name;
   private Object value;
   private ArrayList fields;
}
```

javax.swing.tree.TreeModel

- Object getRoot()

                         .

- int getChildCount(Object parent)
  parent   .
- Object getChild(Object parent, int index)
                         parent   .
- int getIndexOfChild(Object parent, Object child)
  child   . parent   .
- boolean isLeaf(Object   node)
            node   true   .
- void addTreeModelListener(TreeModelListener l)
- void removeTreeModelListener(TreeModelListener l)

                                                                    .

- void valueForPathChanged(TreePath path, Object newValue)

                                                    .

                   :   path

                            object   .

javax.swing.event.TreeModelListener

- void treeNodesChanged(TreeModelEvent e)
- void treeNodesInserted(TreeModelEvent e)
- void treeNodesRemoved(TreeModelEvent e)
- void treeStructureChanged(TreeModelEvent e)

.

javax.swing.event.TreeModelLEvent

- TreeModelEvent(Object eventSource, TreePath node)

.

： eventSource

node

javax.swing.EventListenerList

- void add(Class t, EventListener l)

.

.

addXxxListener ：

public void addXxxListener(XxxListener l)

{ listenerList.add(Xxx.class, l);

}

: c

l

- void remove(Class t, EventListener l)

.

addXxxListener ：

public void removeXxxListener(XxxListener l)

{ listenerList.remove(XxxListener.class, l);

}

: c

l

- Object[] getListenerList()

. .

.

```
protected void fireEventOccured(EventParameters p)
{    XxxEvent event = new XxxEvent(this , p);
     Object[] listeners = listenerList.getListenerList();
     for(int i = listeners.length –2; i >- 0; i -= 2)
```

```
if(listeners[i] instanceof XxxListener.class)
 ((XxxListener)listeners[i+1]).
             EventOccured(event);
```