Outline

• Trees
• Program
**Trees Overview**

- **JTree** is a component that displays data in a treelike hierarchy
- All nodes displayed in the tree are in the form of a hierarchical indexed list
  - A node can have child nodes
  - A node with no children is a leaf
  - A node with no parent is the root
- **TreeModel** represents the entire tree
- **TreeNode** represents a node
- **TreePath** represents a path to a node
- **TreeModel** does not directly store or manage tree data (different from the **TableModel** with JTables)
- **MutableTreeNode** represents a tree node that can be mutated by adding or removing child nodes or changing the content of the node
- **TreeSelectionModel** handles the tree node selection
JTree Hierarchy

```
javax.swing.JTree
  -model: TreeModel
  -anchorPath: TreePath
  -leadPath: TreePath
  -selectionModel: TreeSelectionModel
  -cellRenderer: TreeCellRenderer
  -cellEditor: TreeCellEditor

TreeNode
  -DefaultTreeNode
  -MutableTreeNode
    -DefaultMutableTreeNode
      -TreeModel
        -DefaultTreeModel
      -TreePath
      -TreeSelectionModel
        -DefaultTreeSelectionModel
      -TreeCellRenderer
        -DefaultTreeCellRenderer
      -TreeCellEditor
        -DefaultTreeCellEditor
```
JTree API

```
import javax.swing.JTree

public class JTree {
    public JTree() {
    }
    public JTree(value: java.util.Hashtable) {
    }
    public JTree(value: Object[]) {
    }
    public JTree(newModel: TreeModel) {
    }
    public JTree(root: TreeNode) {
    }
    public JTree(root: TreeNode, asksAllowsChildren: boolean) {
    }
    public JTree(value: Vector) {
    }
    public void addSelectionPath(path: TreePath) {
    }
    public void addSelectionPaths(paths: TreePath[]) {
    }
    public void addSelectionRow(row: int) {
    }
    public void addSelectionRows(rows: int[]) {
    }
    public void clearSelection() {
    }
    public void collapsePath(path: TreePath) {
    }
    public TreePath getSelectionPath() {
    }
    public TreePath getSelectionPaths() {
    }
    public TreePath getLastSelectedPathComponent() {
    }
    public int getRowCount() {
    }
    public void removeSelectionPath(path: TreePath) {
    }
    public void removeSelectionPaths(paths: TreePath[]) {
    }
}
```

JavaBeans properties with get and set methods omitted in the UML diagram.

- `cellEditor`: Specifies a cell editor used to edit entries in the tree.
- `cellRenderer`: Specifies a cell renderer.
- `editable`: Specifies whether the cells are editable (default: false).
- `model`: Maintains the tree model.
- `rootVisible`: Specifies whether the root is displayed (depending on the constructor).
- `rowHeight`: Specifies the height of the row for the node displayed in the tree (default: 16 pixels).
- `scrollsOnExpand`: If true, when a node is expanded, as many of the descendants as possible are scrolled to be visible (default: true).
- `selectModel`: Models the set of selected nodes in this tree.
- `showRootHandles`: Specifies whether the root handles are displayed (default: true).
- `toggleClickCount`: Number of mouse clicks before a node is expanded (default: 2).
- `anchorSelectionPath`: The path identified as the anchor.
- `exandsSelectedPaths`: True if paths in the selection should be expanded (default: true).
- `leadSelectionPath`: The path identified as the lead.

Creates a JTree with a sample tree model, as shown in Figure 53.18.

Creates a JTree with an invisible root and the keys in the Hashtable key/value pairs as its children.

Creates a JTree with an invisible root and the elements in the array as its children.

Creates a JTree with the specified tree model.

Creates a JTree with the specified tree node as its root.

Creates a JTree with the specified tree node as its root and decides whether a node is a leaf node in the specified manner.

Creates a JTree with an invisible root and the elements in the vector as its children.

Adds the specified TreePath to the current selection.

Adds the specified TreePaths to the current selection.

Adds the path at the specified row to the current selection.

Adds the path at the specified row to the current selection.

Clears the selection.

Ensures that the node identified by the specified path is collapsed and viewable.

Returns the path from the root to the first selected node.

Returns the paths from the root to all the selected nodes.

Returns the last node in the first selected TreePath.

Returns the number of rows currently being displayed.

Removes the node in the specified path.

Removes the node in the specified path.
```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTree;

public class Test extends JFrame {
    public Test() {
        super("Tree Example");

        JPanel jp = new JPanel();
        String [] data = {"Cat", "Dog", "Fish", "Frog", "Salamander");

        JTree tree = new JTree(data);
        tree.setRootVisible(true);
        JScrollPane jsp = new JScrollPane(tree);
        jp.add(jsp);
        add(jp);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 200);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String args[]) {
        Test t = new Test();
    }
}
```
Tree Model Example

```java
// import statements omitted

public class Test extends JFrame {
    public Test() {
        super("Tree Example");

        JPanel jp = new JPanel();
        DefaultMutableTreeNode root, europe, northAmerica, us;
        europe = new DefaultMutableTreeNode("Europe");
        europe.add(new DefaultMutableTreeNode("UK");
        europe.add(new DefaultMutableTreeNode("Germany");
        europe.add(new DefaultMutableTreeNode("France");
        europe.add(new DefaultMutableTreeNode("Norway");

        northAmerica = new DefaultMutableTreeNode("North America");
        us = new DefaultMutableTreeNode("US");
        us.add(new DefaultMutableTreeNode("California");
        us.add(new DefaultMutableTreeNode("Texas");
        us.add(new DefaultMutableTreeNode("New York");
        us.add(new DefaultMutableTreeNode("Florida");
        us.add(new DefaultMutableTreeNode("Illinois");
        northAmerica.add(us);
        northAmerica.add(new DefaultMutableTreeNode("Canada");

        root = new DefaultMutableTreeNode("World");
        root.add(europe);
        root.add(northAmerica);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 2));
        panel.add(new JScrollPane(new JTree(root)));
        panel.add(new JScrollPane(new JTree(new DefaultTreeModel(root))));

        JTextArea jtaMessage = new JTextArea();
        jtaMessage.setWrapStyleWord(true);
        jtaMessage.setLineWrap(true);
        JScrollPane jtaScrollPane = new JScrollPane(jtaMessage);
        add(jtaScrollPane, BorderLayout.CENTER);

        // Get tree information
        jtaMessage.append("Depth of the node US is " + us.getDepth());
        jtaMessage.append("Level of the node US is " + us.getLevel());
        jtaMessage.append("First child of the root is " + root.getFirstChild());
        jtaMessage.append("First leaf of the root is " + root.getFirstLeaf());
        jtaMessage.append("Number of children of the root is " + root.getChildCount());
        jtaMessage.append("Number of leaves in the tree is " + root.getLeafCount());
        String breadthFirstSearchResult = "";

        // Breadth-first traversal
        Enumeration bf = root.breadthFirstEnumeration();
        while (bf.hasMoreElements()) {
            breadthFirstSearchResult += bf.nextElement().toString() + " ";
        }
        jtaMessage.append("Breath-first traversal from the root is " + breadthFirstSearchResult);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 200);
        setLocationRelativeTo(null);
        setVisible(true);  
    }

    public static void main(String args[]) {
        Test t = new Test();
    }
}
```
More Tree Details

- **JTree can fire a TreeSelectionEvent and a TreeExpansionEvent**
  - When a new node is selected, a **TreeSelectionEvent** is fired
    - To handle the **TreeSelectionEvent**, a listener must implement the **TreeSelectionListener** interface
  - When a node is expanded or collapsed, a **TreeExpansionEvent** is fired
    - To handle the **TreeExpansionEvent**, a listener must implement the **TreeExpansionListener** interface (or extend **TreeExpansionAdapter**)

USC Viterbi
School of Engineering
Outline

• Trees
• Program
Program

- Create the following GUI.