| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DefaultMutableTreeNode.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/DefaultTreeCellEditor.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/DefaultMutableTreeNode.html)    [**NO FRAMES**](http://docs.google.com/DefaultMutableTreeNode.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#lnxbz9) | [METHOD](#2jxsxqh) |

## **javax.swing.tree**

Class DefaultMutableTreeNode

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.tree.DefaultMutableTreeNode**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html), [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Direct Known Subclasses:** [JTree.DynamicUtilTreeNode](http://docs.google.com/javax/swing/JTree.DynamicUtilTreeNode.html)

public class **DefaultMutableTreeNode**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

A DefaultMutableTreeNode is a general-purpose node in a tree data structure. For examples of using default mutable tree nodes, see [How to Use Trees](http://java.sun.com/docs/books/tutorial/uiswing/components/tree.html) in *The Java Tutorial.*

A tree node may have at most one parent and 0 or more children. DefaultMutableTreeNode provides operations for examining and modifying a node's parent and children and also operations for examining the tree that the node is a part of. A node's tree is the set of all nodes that can be reached by starting at the node and following all the possible links to parents and children. A node with no parent is the root of its tree; a node with no children is a leaf. A tree may consist of many subtrees, each node acting as the root for its own subtree.

This class provides enumerations for efficiently traversing a tree or subtree in various orders or for following the path between two nodes. A DefaultMutableTreeNode may also hold a reference to a user object, the use of which is left to the user. Asking a DefaultMutableTreeNode for its string representation with toString() returns the string representation of its user object.

**This is not a thread safe class.**If you intend to use a DefaultMutableTreeNode (or a tree of TreeNodes) in more than one thread, you need to do your own synchronizing. A good convention to adopt is synchronizing on the root node of a tree.

While DefaultMutableTreeNode implements the MutableTreeNode interface and will allow you to add in any implementation of MutableTreeNode not all of the methods in DefaultMutableTreeNode will be applicable to all MutableTreeNodes implementations. Especially with some of the enumerations that are provided, using some of these methods assumes the DefaultMutableTreeNode contains only DefaultMutableNode instances. All of the TreeNode/MutableTreeNode methods will behave as defined no matter what implementations are added.

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

**See Also:**[MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html)

| **Field Summary** | |
| --- | --- |
| protected  boolean | [**allowsChildren**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#allowsChildren)            true if the node is able to have children |
| protected  [Vector](http://docs.google.com/java/util/Vector.html) | [**children**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#children)            array of children, may be null if this node has no children |
| static [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)> | [**EMPTY\_ENUMERATION**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#EMPTY_ENUMERATION)            An enumeration that is always empty. |
| protected  [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) | [**parent**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#parent)            this node's parent, or null if this node has no parent |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**userObject**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#userObject)            optional user object |

| **Constructor Summary** | |
| --- | --- |
| [**DefaultMutableTreeNode**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#DefaultMutableTreeNode())()            Creates a tree node that has no parent and no children, but which allows children. |
| [**DefaultMutableTreeNode**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#DefaultMutableTreeNode(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) userObject)            Creates a tree node with no parent, no children, but which allows children, and initializes it with the specified user object. |
| [**DefaultMutableTreeNode**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#DefaultMutableTreeNode(java.lang.Object,%20boolean))([Object](http://docs.google.com/java/lang/Object.html) userObject, boolean allowsChildren)            Creates a tree node with no parent, no children, initialized with the specified user object, and that allows children only if specified. |

| **Method Summary** | |
| --- | --- |
| void | [**add**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#add(javax.swing.tree.MutableTreeNode))([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newChild)            Removes newChild from its parent and makes it a child of this node by adding it to the end of this node's child array. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html) | [**breadthFirstEnumeration**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#breadthFirstEnumeration())()            Creates and returns an enumeration that traverses the subtree rooted at this node in breadth-first order. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html) | [**children**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#children())()            Creates and returns a forward-order enumeration of this node's children. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#clone())()            Overridden to make clone public. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html) | [**depthFirstEnumeration**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#depthFirstEnumeration())()            Creates and returns an enumeration that traverses the subtree rooted at this node in depth-first order. |
| boolean | [**getAllowsChildren**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getAllowsChildren())()            Returns true if this node is allowed to have children. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getChildAfter**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getChildAfter(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aChild)            Returns the child in this node's child array that immediately follows aChild, which must be a child of this node. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getChildAt**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getChildAt(int))(int index)            Returns the child at the specified index in this node's child array. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getChildBefore**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getChildBefore(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aChild)            Returns the child in this node's child array that immediately precedes aChild, which must be a child of this node. |
| int | [**getChildCount**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getChildCount())()            Returns the number of children of this node. |
| int | [**getDepth**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getDepth())()            Returns the depth of the tree rooted at this node -- the longest distance from this node to a leaf. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getFirstChild**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getFirstChild())()            Returns this node's first child. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getFirstLeaf**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getFirstLeaf())()            Finds and returns the first leaf that is a descendant of this node -- either this node or its first child's first leaf. |
| int | [**getIndex**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getIndex(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aChild)            Returns the index of the specified child in this node's child array. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getLastChild**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getLastChild())()            Returns this node's last child. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getLastLeaf**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getLastLeaf())()            Finds and returns the last leaf that is a descendant of this node -- either this node or its last child's last leaf. |
| int | [**getLeafCount**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getLeafCount())()            Returns the total number of leaves that are descendants of this node. |
| int | [**getLevel**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getLevel())()            Returns the number of levels above this node -- the distance from the root to this node. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getNextLeaf**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getNextLeaf())()            Returns the leaf after this node or null if this node is the last leaf in the tree. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getNextNode**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getNextNode())()            Returns the node that follows this node in a preorder traversal of this node's tree. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getNextSibling**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getNextSibling())()            Returns the next sibling of this node in the parent's children array. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getParent**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getParent())()            Returns this node's parent or null if this node has no parent. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] | [**getPath**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getPath())()            Returns the path from the root, to get to this node. |
| protected  [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] | [**getPathToRoot**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getPathToRoot(javax.swing.tree.TreeNode,%20int))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode, int depth)            Builds the parents of node up to and including the root node, where the original node is the last element in the returned array. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getPreviousLeaf**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getPreviousLeaf())()            Returns the leaf before this node or null if this node is the first leaf in the tree. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getPreviousNode**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getPreviousNode())()            Returns the node that precedes this node in a preorder traversal of this node's tree. |
| [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) | [**getPreviousSibling**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getPreviousSibling())()            Returns the previous sibling of this node in the parent's children array. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getRoot**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getRoot())()            Returns the root of the tree that contains this node. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**getSharedAncestor**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode))([DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) aNode)            Returns the nearest common ancestor to this node and aNode. |
| int | [**getSiblingCount**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getSiblingCount())()            Returns the number of siblings of this node. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getUserObject**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getUserObject())()            Returns this node's user object. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**getUserObjectPath**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getUserObjectPath())()            Returns the user object path, from the root, to get to this node. |
| void | [**insert**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#insert(javax.swing.tree.MutableTreeNode,%20int))([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newChild, int childIndex)            Removes newChild from its present parent (if it has a parent), sets the child's parent to this node, and then adds the child to this node's child array at index childIndex. |
| boolean | [**isLeaf**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isLeaf())()            Returns true if this node has no children. |
| boolean | [**isNodeAncestor**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeAncestor(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) anotherNode)            Returns true if anotherNode is an ancestor of this node -- if it is this node, this node's parent, or an ancestor of this node's parent. |
| boolean | [**isNodeChild**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeChild(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode)            Returns true if aNode is a child of this node. |
| boolean | [**isNodeDescendant**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode))([DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) anotherNode)            Returns true if anotherNode is a descendant of this node -- if it is this node, one of this node's children, or a descendant of one of this node's children. |
| boolean | [**isNodeRelated**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeRelated(javax.swing.tree.DefaultMutableTreeNode))([DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) aNode)            Returns true if and only if aNode is in the same tree as this node. |
| boolean | [**isNodeSibling**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeSibling(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) anotherNode)            Returns true if anotherNode is a sibling of (has the same parent as) this node. |
| boolean | [**isRoot**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isRoot())()            Returns true if this node is the root of the tree. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html) | [**pathFromAncestorEnumeration**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#pathFromAncestorEnumeration(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) ancestor)            Creates and returns an enumeration that follows the path from ancestor to this node. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html) | [**postorderEnumeration**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#postorderEnumeration())()            Creates and returns an enumeration that traverses the subtree rooted at this node in postorder. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html) | [**preorderEnumeration**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#preorderEnumeration())()            Creates and returns an enumeration that traverses the subtree rooted at this node in preorder. |
| void | [**remove**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#remove(int))(int childIndex)            Removes the child at the specified index from this node's children and sets that node's parent to null. |
| void | [**remove**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#remove(javax.swing.tree.MutableTreeNode))([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) aChild)            Removes aChild from this node's child array, giving it a null parent. |
| void | [**removeAllChildren**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#removeAllChildren())()            Removes all of this node's children, setting their parents to null. |
| void | [**removeFromParent**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#removeFromParent())()            Removes the subtree rooted at this node from the tree, giving this node a null parent. |
| void | [**setAllowsChildren**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#setAllowsChildren(boolean))(boolean allows)            Determines whether or not this node is allowed to have children. |
| void | [**setParent**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#setParent(javax.swing.tree.MutableTreeNode))([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newParent)            Sets this node's parent to newParent but does not change the parent's child array. |
| void | [**setUserObject**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#setUserObject(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) userObject)            Sets the user object for this node to userObject. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#toString())()            Returns the result of sending toString() to this node's user object, or null if this node has no user object. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### EMPTY\_ENUMERATION

public static final [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)> **EMPTY\_ENUMERATION**

An enumeration that is always empty. This is used when an enumeration of a leaf node's children is requested.

### parent

protected [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) **parent**

this node's parent, or null if this node has no parent

### children

protected [Vector](http://docs.google.com/java/util/Vector.html) **children**

array of children, may be null if this node has no children

### userObject

protected transient [Object](http://docs.google.com/java/lang/Object.html) **userObject**

optional user object

### allowsChildren

protected boolean **allowsChildren**

true if the node is able to have children

| **Constructor Detail** |
| --- |

### DefaultMutableTreeNode

public **DefaultMutableTreeNode**()

Creates a tree node that has no parent and no children, but which allows children.

### DefaultMutableTreeNode

public **DefaultMutableTreeNode**([Object](http://docs.google.com/java/lang/Object.html) userObject)

Creates a tree node with no parent, no children, but which allows children, and initializes it with the specified user object.

**Parameters:**userObject - an Object provided by the user that constitutes the node's data

### DefaultMutableTreeNode

public **DefaultMutableTreeNode**([Object](http://docs.google.com/java/lang/Object.html) userObject,  
 boolean allowsChildren)

Creates a tree node with no parent, no children, initialized with the specified user object, and that allows children only if specified.

**Parameters:**userObject - an Object provided by the user that constitutes the node's dataallowsChildren - if true, the node is allowed to have child nodes -- otherwise, it is always a leaf node

| **Method Detail** |
| --- |

### insert

public void **insert**([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newChild,  
 int childIndex)

Removes newChild from its present parent (if it has a parent), sets the child's parent to this node, and then adds the child to this node's child array at index childIndex. newChild must not be null and must not be an ancestor of this node.

**Specified by:**[insert](http://docs.google.com/javax/swing/tree/MutableTreeNode.html#insert(javax.swing.tree.MutableTreeNode,%20int)) in interface [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) **Parameters:**newChild - the MutableTreeNode to insert under this nodechildIndex - the index in this node's child array where this node is to be inserted **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if childIndex is out of bounds [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if newChild is null or is an ancestor of this node [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node does not allow children**See Also:**[isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode))

### remove

public void **remove**(int childIndex)

Removes the child at the specified index from this node's children and sets that node's parent to null. The child node to remove must be a MutableTreeNode.

**Specified by:**[remove](http://docs.google.com/javax/swing/tree/MutableTreeNode.html#remove(int)) in interface [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) **Parameters:**childIndex - the index in this node's child array of the child to remove **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if childIndex is out of bounds

### setParent

public void **setParent**([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newParent)

Sets this node's parent to newParent but does not change the parent's child array. This method is called from insert() and remove() to reassign a child's parent, it should not be messaged from anywhere else.

**Specified by:**[setParent](http://docs.google.com/javax/swing/tree/MutableTreeNode.html#setParent(javax.swing.tree.MutableTreeNode)) in interface [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) **Parameters:**newParent - this node's new parent

### getParent

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getParent**()

Returns this node's parent or null if this node has no parent.

**Specified by:**[getParent](http://docs.google.com/javax/swing/tree/TreeNode.html#getParent()) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Returns:**this node's parent TreeNode, or null if this node has no parent

### getChildAt

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getChildAt**(int index)

Returns the child at the specified index in this node's child array.

**Specified by:**[getChildAt](http://docs.google.com/javax/swing/tree/TreeNode.html#getChildAt(int)) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Parameters:**index - an index into this node's child array **Returns:**the TreeNode in this node's child array at the specified index **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if index is out of bounds

### getChildCount

public int **getChildCount**()

Returns the number of children of this node.

**Specified by:**[getChildCount](http://docs.google.com/javax/swing/tree/TreeNode.html#getChildCount()) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Returns:**an int giving the number of children of this node

### getIndex

public int **getIndex**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aChild)

Returns the index of the specified child in this node's child array. If the specified node is not a child of this node, returns -1. This method performs a linear search and is O(n) where n is the number of children.

**Specified by:**[getIndex](http://docs.google.com/javax/swing/tree/TreeNode.html#getIndex(javax.swing.tree.TreeNode)) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Parameters:**aChild - the TreeNode to search for among this node's children **Returns:**an int giving the index of the node in this node's child array, or -1 if the specified node is a not a child of this node **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if aChild is null

### children

public [Enumeration](http://docs.google.com/java/util/Enumeration.html) **children**()

Creates and returns a forward-order enumeration of this node's children. Modifying this node's child array invalidates any child enumerations created before the modification.

**Specified by:**[children](http://docs.google.com/javax/swing/tree/TreeNode.html#children()) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Returns:**an Enumeration of this node's children

### setAllowsChildren

public void **setAllowsChildren**(boolean allows)

Determines whether or not this node is allowed to have children. If allows is false, all of this node's children are removed.

Note: By default, a node allows children.

**Parameters:**allows - true if this node is allowed to have children

### getAllowsChildren

public boolean **getAllowsChildren**()

Returns true if this node is allowed to have children.

**Specified by:**[getAllowsChildren](http://docs.google.com/javax/swing/tree/TreeNode.html#getAllowsChildren()) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Returns:**true if this node allows children, else false

### setUserObject

public void **setUserObject**([Object](http://docs.google.com/java/lang/Object.html) userObject)

Sets the user object for this node to userObject.

**Specified by:**[setUserObject](http://docs.google.com/javax/swing/tree/MutableTreeNode.html#setUserObject(java.lang.Object)) in interface [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) **Parameters:**userObject - the Object that constitutes this node's user-specified data**See Also:**[getUserObject()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getUserObject()), [toString()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#toString())

### getUserObject

public [Object](http://docs.google.com/java/lang/Object.html) **getUserObject**()

Returns this node's user object.

**Returns:**the Object stored at this node by the user**See Also:**[setUserObject(java.lang.Object)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#setUserObject(java.lang.Object)), [toString()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#toString())

### removeFromParent

public void **removeFromParent**()

Removes the subtree rooted at this node from the tree, giving this node a null parent. Does nothing if this node is the root of its tree.

**Specified by:**[removeFromParent](http://docs.google.com/javax/swing/tree/MutableTreeNode.html#removeFromParent()) in interface [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html)

### remove

public void **remove**([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) aChild)

Removes aChild from this node's child array, giving it a null parent.

**Specified by:**[remove](http://docs.google.com/javax/swing/tree/MutableTreeNode.html#remove(javax.swing.tree.MutableTreeNode)) in interface [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) **Parameters:**aChild - a child of this node to remove **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if aChild is null or is not a child of this node

### removeAllChildren

public void **removeAllChildren**()

Removes all of this node's children, setting their parents to null. If this node has no children, this method does nothing.

### add

public void **add**([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newChild)

Removes newChild from its parent and makes it a child of this node by adding it to the end of this node's child array.

**Parameters:**newChild - node to add as a child of this node **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if newChild is null [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node does not allow children**See Also:**[insert(javax.swing.tree.MutableTreeNode, int)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#insert(javax.swing.tree.MutableTreeNode,%20int))

### isNodeAncestor

public boolean **isNodeAncestor**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) anotherNode)

Returns true if anotherNode is an ancestor of this node -- if it is this node, this node's parent, or an ancestor of this node's parent. (Note that a node is considered an ancestor of itself.) If anotherNode is null, this method returns false. This operation is at worst O(h) where h is the distance from the root to this node.

**Parameters:**anotherNode - node to test as an ancestor of this node **Returns:**true if this node is a descendant of anotherNode**See Also:**[isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)), [getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode))

### isNodeDescendant

public boolean **isNodeDescendant**([DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) anotherNode)

Returns true if anotherNode is a descendant of this node -- if it is this node, one of this node's children, or a descendant of one of this node's children. Note that a node is considered a descendant of itself. If anotherNode is null, returns false. This operation is at worst O(h) where h is the distance from the root to anotherNode.

**Parameters:**anotherNode - node to test as descendant of this node **Returns:**true if this node is an ancestor of anotherNode**See Also:**[isNodeAncestor(javax.swing.tree.TreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeAncestor(javax.swing.tree.TreeNode)), [getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode))

### getSharedAncestor

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getSharedAncestor**([DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) aNode)

Returns the nearest common ancestor to this node and aNode. Returns null, if no such ancestor exists -- if this node and aNode are in different trees or if aNode is null. A node is considered an ancestor of itself.

**Parameters:**aNode - node to find common ancestor with **Returns:**nearest ancestor common to this node and aNode, or null if none**See Also:**[isNodeAncestor(javax.swing.tree.TreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeAncestor(javax.swing.tree.TreeNode)), [isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode))

### isNodeRelated

public boolean **isNodeRelated**([DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) aNode)

Returns true if and only if aNode is in the same tree as this node. Returns false if aNode is null.

**Returns:**true if aNode is in the same tree as this node; false if aNode is null**See Also:**[getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getSharedAncestor(javax.swing.tree.DefaultMutableTreeNode)), [getRoot()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getRoot())

### getDepth

public int **getDepth**()

Returns the depth of the tree rooted at this node -- the longest distance from this node to a leaf. If this node has no children, returns 0. This operation is much more expensive than getLevel() because it must effectively traverse the entire tree rooted at this node.

**Returns:**the depth of the tree whose root is this node**See Also:**[getLevel()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getLevel())

### getLevel

public int **getLevel**()

Returns the number of levels above this node -- the distance from the root to this node. If this node is the root, returns 0.

**Returns:**the number of levels above this node**See Also:**[getDepth()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getDepth())

### getPath

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] **getPath**()

Returns the path from the root, to get to this node. The last element in the path is this node.

**Returns:**an array of TreeNode objects giving the path, where the first element in the path is the root and the last element is this node.

### getPathToRoot

protected [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] **getPathToRoot**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode,  
 int depth)

Builds the parents of node up to and including the root node, where the original node is the last element in the returned array. The length of the returned array gives the node's depth in the tree.

**Parameters:**aNode - the TreeNode to get the path fordepth - an int giving the number of steps already taken towards the root (on recursive calls), used to size the returned array **Returns:**an array of TreeNodes giving the path from the root to the specified node

### getUserObjectPath

public [Object](http://docs.google.com/java/lang/Object.html)[] **getUserObjectPath**()

Returns the user object path, from the root, to get to this node. If some of the TreeNodes in the path have null user objects, the returned path will contain nulls.

### getRoot

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getRoot**()

Returns the root of the tree that contains this node. The root is the ancestor with a null parent.

**Returns:**the root of the tree that contains this node**See Also:**[isNodeAncestor(javax.swing.tree.TreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeAncestor(javax.swing.tree.TreeNode))

### isRoot

public boolean **isRoot**()

Returns true if this node is the root of the tree. The root is the only node in the tree with a null parent; every tree has exactly one root.

**Returns:**true if this node is the root of its tree

### getNextNode

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getNextNode**()

Returns the node that follows this node in a preorder traversal of this node's tree. Returns null if this node is the last node of the traversal. This is an inefficient way to traverse the entire tree; use an enumeration, instead.

**Returns:**the node that follows this node in a preorder traversal, or null if this node is last**See Also:**[preorderEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#preorderEnumeration())

### getPreviousNode

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getPreviousNode**()

Returns the node that precedes this node in a preorder traversal of this node's tree. Returns null if this node is the first node of the traversal -- the root of the tree. This is an inefficient way to traverse the entire tree; use an enumeration, instead.

**Returns:**the node that precedes this node in a preorder traversal, or null if this node is the first**See Also:**[preorderEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#preorderEnumeration())

### preorderEnumeration

public [Enumeration](http://docs.google.com/java/util/Enumeration.html) **preorderEnumeration**()

Creates and returns an enumeration that traverses the subtree rooted at this node in preorder. The first node returned by the enumeration's nextElement() method is this node.

Modifying the tree by inserting, removing, or moving a node invalidates any enumerations created before the modification.

**Returns:**an enumeration for traversing the tree in preorder**See Also:**[postorderEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#postorderEnumeration())

### postorderEnumeration

public [Enumeration](http://docs.google.com/java/util/Enumeration.html) **postorderEnumeration**()

Creates and returns an enumeration that traverses the subtree rooted at this node in postorder. The first node returned by the enumeration's nextElement() method is the leftmost leaf. This is the same as a depth-first traversal.

Modifying the tree by inserting, removing, or moving a node invalidates any enumerations created before the modification.

**Returns:**an enumeration for traversing the tree in postorder**See Also:**[depthFirstEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#depthFirstEnumeration()), [preorderEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#preorderEnumeration())

### breadthFirstEnumeration

public [Enumeration](http://docs.google.com/java/util/Enumeration.html) **breadthFirstEnumeration**()

Creates and returns an enumeration that traverses the subtree rooted at this node in breadth-first order. The first node returned by the enumeration's nextElement() method is this node.

Modifying the tree by inserting, removing, or moving a node invalidates any enumerations created before the modification.

**Returns:**an enumeration for traversing the tree in breadth-first order**See Also:**[depthFirstEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#depthFirstEnumeration())

### depthFirstEnumeration

public [Enumeration](http://docs.google.com/java/util/Enumeration.html) **depthFirstEnumeration**()

Creates and returns an enumeration that traverses the subtree rooted at this node in depth-first order. The first node returned by the enumeration's nextElement() method is the leftmost leaf. This is the same as a postorder traversal.

Modifying the tree by inserting, removing, or moving a node invalidates any enumerations created before the modification.

**Returns:**an enumeration for traversing the tree in depth-first order**See Also:**[breadthFirstEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#breadthFirstEnumeration()), [postorderEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#postorderEnumeration())

### pathFromAncestorEnumeration

public [Enumeration](http://docs.google.com/java/util/Enumeration.html) **pathFromAncestorEnumeration**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) ancestor)

Creates and returns an enumeration that follows the path from ancestor to this node. The enumeration's nextElement() method first returns ancestor, then the child of ancestor that is an ancestor of this node, and so on, and finally returns this node. Creation of the enumeration is O(m) where m is the number of nodes between this node and ancestor, inclusive. Each nextElement() message is O(1).

Modifying the tree by inserting, removing, or moving a node invalidates any enumerations created before the modification.

**Returns:**an enumeration for following the path from an ancestor of this node to this one **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if ancestor is not an ancestor of this node**See Also:**[isNodeAncestor(javax.swing.tree.TreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeAncestor(javax.swing.tree.TreeNode)), [isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode))

### isNodeChild

public boolean **isNodeChild**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode)

Returns true if aNode is a child of this node. If aNode is null, this method returns false.

**Returns:**true if aNode is a child of this node; false if aNode is null

### getFirstChild

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getFirstChild**()

Returns this node's first child. If this node has no children, throws NoSuchElementException.

**Returns:**the first child of this node **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this node has no children

### getLastChild

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getLastChild**()

Returns this node's last child. If this node has no children, throws NoSuchElementException.

**Returns:**the last child of this node **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this node has no children

### getChildAfter

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getChildAfter**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aChild)

Returns the child in this node's child array that immediately follows aChild, which must be a child of this node. If aChild is the last child, returns null. This method performs a linear search of this node's children for aChild and is O(n) where n is the number of children; to traverse the entire array of children, use an enumeration instead.

**Returns:**the child of this node that immediately follows aChild **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if aChild is null or is not a child of this node**See Also:**[children](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#children)

### getChildBefore

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **getChildBefore**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aChild)

Returns the child in this node's child array that immediately precedes aChild, which must be a child of this node. If aChild is the first child, returns null. This method performs a linear search of this node's children for aChild and is O(n) where n is the number of children.

**Returns:**the child of this node that immediately precedes aChild **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if aChild is null or is not a child of this node

### isNodeSibling

public boolean **isNodeSibling**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) anotherNode)

Returns true if anotherNode is a sibling of (has the same parent as) this node. A node is its own sibling. If anotherNode is null, returns false.

**Parameters:**anotherNode - node to test as sibling of this node **Returns:**true if anotherNode is a sibling of this node

### getSiblingCount

public int **getSiblingCount**()

Returns the number of siblings of this node. A node is its own sibling (if it has no parent or no siblings, this method returns 1).

**Returns:**the number of siblings of this node

### getNextSibling

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getNextSibling**()

Returns the next sibling of this node in the parent's children array. Returns null if this node has no parent or is the parent's last child. This method performs a linear search that is O(n) where n is the number of children; to traverse the entire array, use the parent's child enumeration instead.

**Returns:**the sibling of this node that immediately follows this node**See Also:**[children](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#children)

### getPreviousSibling

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getPreviousSibling**()

Returns the previous sibling of this node in the parent's children array. Returns null if this node has no parent or is the parent's first child. This method performs a linear search that is O(n) where n is the number of children.

**Returns:**the sibling of this node that immediately precedes this node

### isLeaf

public boolean **isLeaf**()

Returns true if this node has no children. To distinguish between nodes that have no children and nodes that *cannot* have children (e.g. to distinguish files from empty directories), use this method in conjunction with getAllowsChildren

**Specified by:**[isLeaf](http://docs.google.com/javax/swing/tree/TreeNode.html#isLeaf()) in interface [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **Returns:**true if this node has no children**See Also:**[getAllowsChildren()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getAllowsChildren())

### getFirstLeaf

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getFirstLeaf**()

Finds and returns the first leaf that is a descendant of this node -- either this node or its first child's first leaf. Returns this node if it is a leaf.

**Returns:**the first leaf in the subtree rooted at this node**See Also:**[isLeaf()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isLeaf()), [isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode))

### getLastLeaf

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getLastLeaf**()

Finds and returns the last leaf that is a descendant of this node -- either this node or its last child's last leaf. Returns this node if it is a leaf.

**Returns:**the last leaf in the subtree rooted at this node**See Also:**[isLeaf()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isLeaf()), [isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeDescendant(javax.swing.tree.DefaultMutableTreeNode))

### getNextLeaf

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getNextLeaf**()

Returns the leaf after this node or null if this node is the last leaf in the tree.

In this implementation of the MutableNode interface, this operation is very inefficient. In order to determine the next node, this method first performs a linear search in the parent's child-list in order to find the current node.

That implementation makes the operation suitable for short traversals from a known position. But to traverse all of the leaves in the tree, you should use depthFirstEnumeration to enumerate the nodes in the tree and use isLeaf on each node to determine which are leaves.

**Returns:**returns the next leaf past this node**See Also:**[depthFirstEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#depthFirstEnumeration()), [isLeaf()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isLeaf())

### getPreviousLeaf

public [DefaultMutableTreeNode](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html) **getPreviousLeaf**()

Returns the leaf before this node or null if this node is the first leaf in the tree.

In this implementation of the MutableNode interface, this operation is very inefficient. In order to determine the previous node, this method first performs a linear search in the parent's child-list in order to find the current node.

That implementation makes the operation suitable for short traversals from a known position. But to traverse all of the leaves in the tree, you should use depthFirstEnumeration to enumerate the nodes in the tree and use isLeaf on each node to determine which are leaves.

**Returns:**returns the leaf before this node**See Also:**[depthFirstEnumeration()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#depthFirstEnumeration()), [isLeaf()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isLeaf())

### getLeafCount

public int **getLeafCount**()

Returns the total number of leaves that are descendants of this node. If this node is a leaf, returns 1. This method is O(n) where n is the number of descendants of this node.

**Returns:**the number of leaves beneath this node**See Also:**[isNodeAncestor(javax.swing.tree.TreeNode)](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#isNodeAncestor(javax.swing.tree.TreeNode))

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns the result of sending toString() to this node's user object, or null if this node has no user object.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of the object.**See Also:**[getUserObject()](http://docs.google.com/javax/swing/tree/DefaultMutableTreeNode.html#getUserObject())

### clone

public [Object](http://docs.google.com/java/lang/Object.html) **clone**()

Overridden to make clone public. Returns a shallow copy of this node; the new node has no parent or children and has a reference to the same user object, if any.

**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a copy of this node**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DefaultMutableTreeNode.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/DefaultTreeCellEditor.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/DefaultMutableTreeNode.html)    [**NO FRAMES**](http://docs.google.com/DefaultMutableTreeNode.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#lnxbz9) | [METHOD](#2jxsxqh) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).