| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractLayoutCache.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   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/AbstractLayoutCache.html)    [**NO FRAMES**](http://docs.google.com/AbstractLayoutCache.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#35nkun2) | [METHOD](#44sinio) |

## **javax.swing.tree**

Class AbstractLayoutCache

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

**All Implemented Interfaces:** [RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html) **Direct Known Subclasses:** [FixedHeightLayoutCache](http://docs.google.com/javax/swing/tree/FixedHeightLayoutCache.html), [VariableHeightLayoutCache](http://docs.google.com/javax/swing/tree/VariableHeightLayoutCache.html)

public abstract class **AbstractLayoutCache**extends [Object](http://docs.google.com/java/lang/Object.html)implements [RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html)

**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).

| **Nested Class Summary** | |
| --- | --- |
| static class | [**AbstractLayoutCache.NodeDimensions**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html)            Used by AbstractLayoutCache to determine the size and x origin of a particular node. |

| **Field Summary** | |
| --- | --- |
| protected  [AbstractLayoutCache.NodeDimensions](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) | [**nodeDimensions**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#nodeDimensions)            Object responsible for getting the size of a node. |
| protected  boolean | [**rootVisible**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#rootVisible)            True if the root node is displayed, false if its children are the highest visible nodes. |
| protected  int | [**rowHeight**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#rowHeight)            Height to use for each row. |
| protected  [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) | [**treeModel**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#treeModel)            Model providing information. |
| protected  [TreeSelectionModel](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html) | [**treeSelectionModel**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#treeSelectionModel)            Selection model. |

| **Constructor Summary** | |
| --- | --- |
| [**AbstractLayoutCache**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#AbstractLayoutCache())() |

| **Method Summary** | |
| --- | --- |
| abstract  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getBounds**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getBounds(javax.swing.tree.TreePath,%20java.awt.Rectangle))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) placeIn)            Returns a rectangle giving the bounds needed to draw path. |
| abstract  boolean | [**getExpandedState**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getExpandedState(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Returns true if the path is expanded, and visible. |
| [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) | [**getModel**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getModel())()            Returns the TreeModel that is providing the data. |
| [AbstractLayoutCache.NodeDimensions](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) | [**getNodeDimensions**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getNodeDimensions())()            Returns the object that renders nodes in the tree, and which is responsible for calculating the dimensions of individual nodes. |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getNodeDimensions**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getNodeDimensions(java.lang.Object,%20int,%20int,%20boolean,%20java.awt.Rectangle))([Object](http://docs.google.com/java/lang/Object.html) value, int row, int depth, boolean expanded, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) placeIn)            Returns, by reference in placeIn, the size needed to represent value. |
| abstract  [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) | [**getPathClosestTo**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getPathClosestTo(int,%20int))(int x, int y)            Returns the path to the node that is closest to x,y. |
| abstract  [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) | [**getPathForRow**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getPathForRow(int))(int row)            Returns the path for passed in row. |
| int | [**getPreferredHeight**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getPreferredHeight())()            Returns the preferred height. |
| int | [**getPreferredWidth**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getPreferredWidth(java.awt.Rectangle))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) bounds)            Returns the preferred width for the passed in region. |
| abstract  int | [**getRowCount**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getRowCount())()            Number of rows being displayed. |
| abstract  int | [**getRowForPath**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getRowForPath(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Returns the row that the last item identified in path is visible at. |
| int | [**getRowHeight**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getRowHeight())()            Returns the height of each row. |
| int[] | [**getRowsForPaths**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getRowsForPaths(javax.swing.tree.TreePath%5B%5D))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)            Returns the rows that the TreePath instances in path are being displayed at. |
| [TreeSelectionModel](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html) | [**getSelectionModel**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getSelectionModel())()            Returns the model used to maintain the selection. |
| abstract  int | [**getVisibleChildCount**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getVisibleChildCount(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Returns the number of visible children for row. |
| abstract  [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)> | [**getVisiblePathsFrom**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#getVisiblePathsFrom(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Returns an Enumerator that increments over the visible paths starting at the passed in location. |
| abstract  void | [**invalidatePathBounds**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#invalidatePathBounds(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Instructs the LayoutCache that the bounds for path are invalid, and need to be updated. |
| abstract  void | [**invalidateSizes**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#invalidateSizes())()            Informs the TreeState that it needs to recalculate all the sizes it is referencing. |
| abstract  boolean | [**isExpanded**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#isExpanded(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Returns true if the value identified by row is currently expanded. |
| protected  boolean | [**isFixedRowHeight**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#isFixedRowHeight())()            Returns true if the height of each row is a fixed size. |
| boolean | [**isRootVisible**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#isRootVisible())()            Returns true if the root node of the tree is displayed. |
| abstract  void | [**setExpandedState**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#setExpandedState(javax.swing.tree.TreePath,%20boolean))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path, boolean isExpanded)            Marks the path path expanded state to isExpanded. |
| void | [**setModel**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#setModel(javax.swing.tree.TreeModel))([TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) newModel)            Sets the TreeModel that will provide the data. |
| void | [**setNodeDimensions**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#setNodeDimensions(javax.swing.tree.AbstractLayoutCache.NodeDimensions))([AbstractLayoutCache.NodeDimensions](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) nd)            Sets the renderer that is responsible for drawing nodes in the tree and which is threfore responsible for calculating the dimensions of individual nodes. |
| void | [**setRootVisible**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#setRootVisible(boolean))(boolean rootVisible)            Determines whether or not the root node from the TreeModel is visible. |
| void | [**setRowHeight**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#setRowHeight(int))(int rowHeight)            Sets the height of each cell. |
| void | [**setSelectionModel**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#setSelectionModel(javax.swing.tree.TreeSelectionModel))([TreeSelectionModel](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html) newLSM)            Sets the TreeSelectionModel used to manage the selection to new LSM. |
| abstract  void | [**treeNodesChanged**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#treeNodesChanged(javax.swing.event.TreeModelEvent))([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)             Invoked after a node (or a set of siblings) has changed in some way. |
| abstract  void | [**treeNodesInserted**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#treeNodesInserted(javax.swing.event.TreeModelEvent))([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)            Invoked after nodes have been inserted into the tree. |
| abstract  void | [**treeNodesRemoved**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#treeNodesRemoved(javax.swing.event.TreeModelEvent))([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)            Invoked after nodes have been removed from the tree. |
| abstract  void | [**treeStructureChanged**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#treeStructureChanged(javax.swing.event.TreeModelEvent))([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)            Invoked after the tree has drastically changed structure from a given node down. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [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()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [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** |
| --- |

### nodeDimensions

protected [AbstractLayoutCache.NodeDimensions](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) **nodeDimensions**

Object responsible for getting the size of a node.

### treeModel

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

Model providing information.

### treeSelectionModel

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

Selection model.

### rootVisible

protected boolean **rootVisible**

True if the root node is displayed, false if its children are the highest visible nodes.

### rowHeight

protected int **rowHeight**

Height to use for each row. If this is <= 0 the renderer will be used to determine the height for each row.

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

### AbstractLayoutCache

public **AbstractLayoutCache**()

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

### setNodeDimensions

public void **setNodeDimensions**([AbstractLayoutCache.NodeDimensions](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) nd)

Sets the renderer that is responsible for drawing nodes in the tree and which is threfore responsible for calculating the dimensions of individual nodes.

**Parameters:**nd - a NodeDimensions object

### getNodeDimensions

public [AbstractLayoutCache.NodeDimensions](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) **getNodeDimensions**()

Returns the object that renders nodes in the tree, and which is responsible for calculating the dimensions of individual nodes.

**Returns:**the NodeDimensions object

### setModel

public void **setModel**([TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) newModel)

Sets the TreeModel that will provide the data.

**Parameters:**newModel - the TreeModel that is to provide the data

### getModel

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

Returns the TreeModel that is providing the data.

**Returns:**the TreeModel that is providing the data

### setRootVisible

public void **setRootVisible**(boolean rootVisible)

Determines whether or not the root node from the TreeModel is visible.

**Parameters:**rootVisible - true if the root node of the tree is to be displayed**See Also:**[rootVisible](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#rootVisible)

### isRootVisible

public boolean **isRootVisible**()

Returns true if the root node of the tree is displayed.

**Returns:**true if the root node of the tree is displayed**See Also:**[rootVisible](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.html#rootVisible)

### setRowHeight

public void **setRowHeight**(int rowHeight)

Sets the height of each cell. If the specified value is less than or equal to zero the current cell renderer is queried for each row's height.

**Parameters:**rowHeight - the height of each cell, in pixels

### getRowHeight

public int **getRowHeight**()

Returns the height of each row. If the returned value is less than or equal to 0 the height for each row is determined by the renderer.

### setSelectionModel

public void **setSelectionModel**([TreeSelectionModel](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html) newLSM)

Sets the TreeSelectionModel used to manage the selection to new LSM.

**Parameters:**newLSM - the new TreeSelectionModel

### getSelectionModel

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

Returns the model used to maintain the selection.

**Returns:**the treeSelectionModel

### getPreferredHeight

public int **getPreferredHeight**()

Returns the preferred height.

**Returns:**the preferred height

### getPreferredWidth

public int **getPreferredWidth**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) bounds)

Returns the preferred width for the passed in region. The region is defined by the path closest to (bounds.x, bounds.y) and ends at bounds.height + bounds.y. If bounds is null, the preferred width for all the nodes will be returned (and this may be a VERY expensive computation).

**Parameters:**bounds - the region being queried **Returns:**the preferred width for the passed in region

### isExpanded

public abstract boolean **isExpanded**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Returns true if the value identified by row is currently expanded.

### getBounds

public abstract [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getBounds**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) placeIn)

Returns a rectangle giving the bounds needed to draw path.

**Parameters:**path - a TreePath specifying a nodeplaceIn - a Rectangle object giving the available space **Returns:**a Rectangle object specifying the space to be used

### getPathForRow

public abstract [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) **getPathForRow**(int row)

Returns the path for passed in row. If row is not visible null is returned.

**Parameters:**row - the row being queried **Returns:**the TreePath for the given row

### getRowForPath

public abstract int **getRowForPath**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Returns the row that the last item identified in path is visible at. Will return -1 if any of the elements in path are not currently visible.

**Parameters:**path - the TreePath being queried **Returns:**the row where the last item in path is visible or -1 if any elements in path aren't currently visible

### getPathClosestTo

public abstract [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) **getPathClosestTo**(int x,  
 int y)

Returns the path to the node that is closest to x,y. If there is nothing currently visible this will return null, otherwise it'll always return a valid path. If you need to test if the returned object is exactly at x, y you should get the bounds for the returned path and test x, y against that.

**Parameters:**x - the horizontal component of the desired locationy - the vertical component of the desired location **Returns:**the TreePath closest to the specified point

### getVisiblePathsFrom

public abstract [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)> **getVisiblePathsFrom**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Returns an Enumerator that increments over the visible paths starting at the passed in location. The ordering of the enumeration is based on how the paths are displayed. The first element of the returned enumeration will be path, unless it isn't visible, in which case null will be returned.

**Parameters:**path - the starting location for the enumeration **Returns:**the Enumerator starting at the desired location

### getVisibleChildCount

public abstract int **getVisibleChildCount**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Returns the number of visible children for row.

**Parameters:**path - the path being queried **Returns:**the number of visible children for the specified path

### setExpandedState

public abstract void **setExpandedState**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path,  
 boolean isExpanded)

Marks the path path expanded state to isExpanded.

**Parameters:**path - the path being expanded or collapsedisExpanded - true if the path should be expanded, false otherwise

### getExpandedState

public abstract boolean **getExpandedState**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Returns true if the path is expanded, and visible.

**Parameters:**path - the path being queried **Returns:**true if the path is expanded and visible, false otherwise

### getRowCount

public abstract int **getRowCount**()

Number of rows being displayed.

**Returns:**the number of rows being displayed

### invalidateSizes

public abstract void **invalidateSizes**()

Informs the TreeState that it needs to recalculate all the sizes it is referencing.

### invalidatePathBounds

public abstract void **invalidatePathBounds**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Instructs the LayoutCache that the bounds for path are invalid, and need to be updated.

**Parameters:**path - the path being updated

### treeNodesChanged

public abstract void **treeNodesChanged**([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)

Invoked after a node (or a set of siblings) has changed in some way. The node(s) have not changed locations in the tree or altered their children arrays, but other attributes have changed and may affect presentation. Example: the name of a file has changed, but it is in the same location in the file system.

e.path() returns the path the parent of the changed node(s).

e.childIndices() returns the index(es) of the changed node(s).

**Parameters:**e - the TreeModelEvent

### treeNodesInserted

public abstract void **treeNodesInserted**([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)

Invoked after nodes have been inserted into the tree.

e.path() returns the parent of the new nodes

e.childIndices() returns the indices of the new nodes in ascending order.

**Parameters:**e - the TreeModelEvent

### treeNodesRemoved

public abstract void **treeNodesRemoved**([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)

Invoked after nodes have been removed from the tree. Note that if a subtree is removed from the tree, this method may only be invoked once for the root of the removed subtree, not once for each individual set of siblings removed.

e.path() returns the former parent of the deleted nodes.

e.childIndices() returns the indices the nodes had before they were deleted in ascending order.

**Parameters:**e - the TreeModelEvent

### treeStructureChanged

public abstract void **treeStructureChanged**([TreeModelEvent](http://docs.google.com/javax/swing/event/TreeModelEvent.html) e)

Invoked after the tree has drastically changed structure from a given node down. If the path returned by e.getPath() is of length one and the first element does not identify the current root node the first element should become the new root of the tree.

e.path() holds the path to the node.

e.childIndices() returns null.

**Parameters:**e - the TreeModelEvent

### getRowsForPaths

public int[] **getRowsForPaths**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)

Returns the rows that the TreePath instances in path are being displayed at. This method should return an array of the same length as that passed in, and if one of the TreePaths in path is not valid its entry in the array should be set to -1.

**Specified by:**[getRowsForPaths](http://docs.google.com/javax/swing/tree/RowMapper.html#getRowsForPaths(javax.swing.tree.TreePath%5B%5D)) in interface [RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html) **Parameters:**paths - the array of TreePaths being queried **Returns:**an array of the same length that is passed in containing the rows that each corresponding where each TreePath is displayed; if paths is null, null is returned

### getNodeDimensions

protected [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getNodeDimensions**([Object](http://docs.google.com/java/lang/Object.html) value,  
 int row,  
 int depth,  
 boolean expanded,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) placeIn)

Returns, by reference in placeIn, the size needed to represent value. If inPlace is null, a newly created Rectangle should be returned, otherwise the value should be placed in inPlace and returned. This will return null if there is no renderer.

**Parameters:**value - the value to be representedrow - row being querieddepth - the depth of the rowexpanded - true if row is expanded, false otherwiseplaceIn - a Rectangle containing the size needed to represent value **Returns:**a Rectangle containing the node dimensions, or null if node has no dimension

### isFixedRowHeight

protected boolean **isFixedRowHeight**()

Returns true if the height of each row is a fixed size.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractLayoutCache.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   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/AbstractLayoutCache.NodeDimensions.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/AbstractLayoutCache.html)    [**NO FRAMES**](http://docs.google.com/AbstractLayoutCache.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#35nkun2) | [METHOD](#44sinio) |

[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).