Class BSTree<E>

java.lang.Object
    BSTree<E>

All Implemented Interfaces:

`java.lang.Cloneable`

```
public class BSTree<E>
extends java.lang.Object
implements java.lang.Cloneable
```

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
`class`	`BSTree.CountRangeVisitor` Visitor for counting items in a given range.
`class`	`BSTree.Node`

Constructor Summary

Constructors

Constructor and Description
`BSTree(java.util.Comparator<E> compare)`
`BSTree(E[] items, java.util.Comparator<E> comp)` Creates the tree from the given preorder array of items

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`void`	`add(E item)` Adds the given item to the tree
`java.lang.Object`	`clone()`
`boolean`	`contains(E item)` Determines if the tree contains the given item
`boolean`	`containsLoop(E item)` Determines if the tree contains the given item
`boolean`	`equals(BSTree.Node root1, BSTree.Node root2)` Compare the 2 given nodes
`boolean`	`equals(java.lang.Object other)` Compare this tree to other object
`BSTree.Node`	`getRoot()` Return the position of the root
`void`	`inorder(Visitor<E> visitor)` Performs inorder traversal of the tree
`boolean`	`isEmpty()` Determine if the tree is empty
`E`	`maxValue()` Return the largest value of the tree
`E`	`maxValueLoop()` Determine the largest value of the tree
`void`	`postorder(Visitor<E> visitor)` Performs postorder traversal of the tree
`void`	`preorder(Visitor<E> visitor)` Performs preorder traversal of the tree
`BSTree.Node`	`rebuildPreorder(E[] items, int i, int j)` Creates a tree from the given preorder array of items
`boolean`	`remove(E item)` Removes the given item from the tree.

`java.lang.String`	`toString()` Return the string representation of the tree level-by-level and from left-to-right

Methods inherited from class java.lang.Object

`getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

BSTree

```
public BSTree(java.util.Comparator<E> compare)
```

BSTree

```
public BSTree(E[] items,
              java.util.Comparator<E> comp)
```

Creates the tree from the given preorder array of items

Parameters:

`items` - An array of items to be inserted into the tree

`comp` - A comparator to define the order of the items

Method Detail

getRoot

```
public BSTree.Node getRoot()
```

Return the position of the root

Returns:

the position of the root

isEmpty

```
public boolean isEmpty()
```

Determine if the tree is empty

Returns:

`true if the tree is empty`

maxValueLoop

```
public E maxValueLoop()
               throws java.util.NoSuchElementException
```

Determine the largest value of the tree

Returns:

`the largest value of the tree`

Throws:

`java.util.NoSuchElementException`

containsLoop

```
public boolean containsLoop(E item)
```

Determines if the tree contains the given item

Parameters:

`item - the given item to see if the tree does contain`

Returns:

`true if the tree contain the item`

add

```
public void add(E item)
         throws java.util.NoSuchElementException
```

Adds the given item to the tree

Parameters:

`item - the given item to be added to the tree`

Throws:

`java.util.NoSuchElementException`

maxValue

```
public E maxValue()
           throws java.util.NoSuchElementException
```

Return the largest value of the tree

Returns:

`the largest value in the tree`

Throws:

`java.util.NoSuchElementException`

contains

```
public boolean contains(E item)
```

Determines if the tree contains the given item

Parameters:

`item - the item to be determined if the tree contains`

Returns:

`true if item found`

remove

```
public boolean remove(E item)
```

Removes the given item from the tree.

Parameters:

`item - given item to be removed`

Returns:

`true if the item is removed.`

preorder

```
public void preorder(Visitor<E> visitor)
```

Performs preorder traversal of the tree

Parameters:

visitor - given visitor to start traverse preorder through

inorder

```
public void inorder(Visitor<E> visitor)
```

Performs inorder traversal of the tree

Parameters:

visitor - given visitor to start traverse inorder through

postorder

```
public void postorder(Visitor<E> visitor)
```

Performs postorder traversal of the tree

Parameters:

visitor - given visitor to start traverse postorder through

equals

```
public boolean equals(java.lang.Object other)
```

Compare this tree to other object

Overrides:

equals in class java.lang.Object

Returns:

true if the given object is equal to the tree

equals

```
public boolean equals(BSTree.Node root1,
                      BSTree.Node root2)
```

Compare the 2 given nodes

Parameters:

root1 - the given node 1

root2 - the given node 2

Returns:

`true if the trees rooted at the given nodes are identical`

clone

```
public java.lang.Object clone()
```

Overrides:

`clone in class java.lang.Object`

Returns:

`a copy of this tree`

rebuildPreorder

```
public BSTree.Node rebuildPreorder(E[] items,
                                   int i,
                                   int j)
```

Creates a tree from the given preorder array of items

Parameters:

`items - The array of items in preorder`

`i - The starting index in the array`

`j - The ending index in the array`

Returns:

`The root node of the rebuilt subtree`

toString

```
public java.lang.String toString()
```

Return the string representation of the tree level-by-level and from left-to-right

Overrides:

`toString in class java.lang.Object`

Returns:

`the string representation of the tree`