Class BTree<E>

java.lang.Object
    BTree<E>

```
public class BTree<E>
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
BTree(int order, java.util.Comparator<E> comp) creates an empty tree with the given order and given comparator

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	add(E item) add the given item to the tree
boolean	contains(E item)
void	inorder(Visitor<E> visitor) perform inorder traversal of the tree
boolean	isEmpty()
boolean	isValid()
java.lang.String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

BTree

```
public BTree(int order,
             java.util.Comparator<E> comp)
```

creates an empty tree with the given order and given comparator

Parameters:

order -

comp -

Method Detail

isEmpty

```
public boolean isEmpty()
```

Returns:

whether the tree is empty

add

```
public void add(E item)
```

add the given item to the tree

Parameters:

item -

contains

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

Parameters:

item -

Returns:

whether the tree contains the given item

inorder

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

perform inorder traversal of the tree

Parameters:

visitor -

toString

public java.lang.String toString()

Overrides:

toString in class java.lang.Object

isValid

public boolean isValid()

Returns:

whether the tree is valid

Class Node<E>

java.lang.Object
    Node<E>

---

public class Node<E>
extends java.lang.Object

Field Summary

Fields

Modifier and Type	Field and Description
java.util.LinkedList<Node>	children
java.util.LinkedList<Node>	data
int	order
Node	parent

Constructor Summary

Constructors

Constructor and Description
Node(int order, java.util.Comparator<E> comp) create an empty node with given order and comparator
Node(int order, java.util.Comparator<E> comp, Node<E> left, E item, Node<E> right) create a node with given left and right children, given comparator and given item
Node(int order, java.util.Comparator<E> comp, Node<E> p, java.util.LinkedList<E> data, java.util.LinkedList<Node<E>> children) create a node with the given parent, given data and given children

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description

void	add(E item) insert the given item in tree(should use for leaf nodes only).
boolean	contains(E item)
boolean	hasOverflow()
boolean	isEmpty()
boolean	isLeaf()
boolean	isValid()
Node<E>	nextChild(E item)
void	split() split the node by creating a new sibling node with half of the items and children of the original node

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

order

public int order

children

public java.util.LinkedList<Node> children

data

public java.util.LinkedList<Node> data

parent

public Node parent

Constructor Detail

Node

```
public  Node(int  order,
             java.util.Comparator<E>  comp)
```

create an empty node with given order and comparator

Parameters:

order -

comp -

Node

```
public  Node(int  order,
             java.util.Comparator<E>  comp,
             Node<E>  left,
             E  item,
             Node<E>  right)
```

create a node with given left and right children, given comparator and given item

Parameters:

order -

comp -

left -

item -

right -

Node

```
public  Node(int  order,
             java.util.Comparator<E>  comp,
             Node<E>  p,
             java.util.LinkedList<E>  data,
             java.util.LinkedList<Node<E>>  children)
```

create a node with the given parent, given data and given children

Parameters:

order -

comp -

p -

data -

children -

Method Detail

hasOverflow

```
public boolean hasOverflow()
```

Returns:

whether the node is filled with number larger than order.

isEmpty

```
public boolean isEmpty()
```

Returns:

whether the node is empty

isLeaf

```
public boolean isLeaf()
```

Returns:

whether the node is leaf, i.e. the node have no child.

nextChild

```
public Node<E> nextChild(E item)
```

Parameters:

item -

Returns:

the next child to follow in order to locate the given item

add

```
public void add(E item)
```

insert the given item in tree(should use for leaf nodes only).

Parameters:

item -

split

```
public void split()
```

split the node by creating a new sibling node with half of the items and children of the original node

contains

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

Parameters:

item -

Returns:

whether the node contains the given item

isValid

```
public boolean isValid()
```

Returns:

whether a node has its data sorted in order and the children have correct link to parents.