Class BTreeNode<E>

java.lang.Object
    BTreeNode<E>

---

public class **BTreeNode<E>**
extends java.lang.Object

Constructor Summary

Constructors

Constructor and Description
BTreeNode(BTreeNode<E> p, java.util.Comparator<E> cmp, int deg)

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
BTreeNode<E>	add(E k) adds k to this node's key list in the right place returns null if there is no overflow after adding other wise:
void	addChild(int index, BTreeNode<E> newChild) inserts new child at index if index is valid for insertion if index invalid, do nothing
void	addFirstChild(BTreeNode<E> newChild) add child to the begininng of the list
void	addFirstKey(E value) inserts value to start of keys list
void	addKey(int index, E value) inserts value at index if index is valid else do nothing
void	addLastChild(BTreeNode<E> newChild) add child to the last of the list
void	addLastKey(E value)

	inserts value to end of keys list
boolean	**contains**(**E** target) returns true if this node has target in its keys
int	**findInsertPos**(**E** new_key) returns the index at which new_key should be inserted in keys arraylist
BTreeNode<**E**>	**getChild**(int index) returns the child of this node at index, if no child at index, return null
BTreeNode<**E**>	**getFirstChild**() returns the first child of this node null otherwise
E	**getFirstKey**() returns the key at index
E	**getKey**(int index) returns the key at index
BTreeNode<**E**>	**getLastChild**() returns the last child of this node null otherwise
E	**getLastKey**() returns the last key of this node null otherwise
int	**indexOf**(**BTreeNode**<**E**> child) returns the index in which child is found in this node's children arraylist or -1 if not found
int	**indexOf**(**E** target) returns the index in which target is found in this node's key arraylist or -1 if not found
boolean	**isEmpty**() returns true if there is no key in this node
boolean	**isLeaf**() returns true if this node is a leaf node
boolean	**isOverflow**() returns true if there is an overflow in this node
static void	**main**(java.lang.String[] args)
int	**nChildren**() returns the number of children
int	**nKeys**() returns the number of keys

BTreeNode<E>	**removeChild**(int index) removes and returns child at index
BTreeNode<E>	**removeFirstChild**() removes and return first child
E	**removeFirstKey**() removes and returns first key
E	**removeKey**(int index) removes and returns key at index
BTreeNode<E>	**removeLastChild**() removes and return last child
E	**removeLastKey**() removes and returns the last key
void	**setKey**(int index, E value) replaces the key at index with value if index is valid else do nothing
java.lang.String	**toString**() returns the data(keys) of this node as a String

Methods inherited from class java.lang.Object

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

Constructor Detail

BTreeNode

```
public BTreeNode(BTreeNode<E> p,
                 java.util.Comparator<E> cmp,
                 int deg)
```

Method Detail

toString

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

returns the data(keys) of this node as a String

Overrides:

toString in class java.lang.Object

Returns:

returns the data(keys) of this node as a String

isLeaf

```
public boolean isLeaf()
```

returns true if this node is a leaf node

Returns:

returns true if this node is a leaf node

isEmpty

```
public boolean isEmpty()
```

returns true if there is no key in this node

Returns:

returns true if there is no key in this node

getChild

```
public BTreeNode<E> getChild(int index)
```

returns the child of this node at index, if no child at index, return null

Parameters:

index -

Returns:

returns the child of this node at index, if no child at index, return null

getFirstChild

```
public BTreeNode<E> getFirstChild()
```

returns the first child of this node null otherwise

Returns:

returns the first child of this node null otherwise

getLastChild

```
public BTreeNode<E> getLastChild()
```

returns the last child of this node null otherwise

Returns:

returns the last child of this node null otherwise

nKeys

```
public int nKeys()
```

returns the number of keys

Returns:

returns the number of keys

nChildren

```
public int nChildren()
```

returns the number of children

Returns:

returns the number of children

getKey

```
public E getKey(int index)
```

returns the key at index

Parameters:

index -

Returns:

returns the key at index

getFirstKey

```
public E getFirstKey()
```

returns the key at index

Parameters:

index -

Returns:

returns the key at index

getLastKey

```
public E getLastKey()
```

returns the last key of this node null otherwise

Returns:

returns the last key of this node null otherwise

addChild

```
public void addChild(int index,
                     BTreeNode<E> newChild)
```

inserts new child at index if index is valid for insertion if index invalid, do nothing

Parameters:

index -

newChild -

addFirstChild

```
public void addFirstChild(BTreeNode<E> newChild)
```

add child to the begininng of the list

Parameters:

newChild -

addLastChild

```
public void addLastChild(BTreeNode<E> newChild)
```

add child to the last of the list

Parameters:

newChild -

setKey

```
public void setKey(int index,
                   E value)
```

replaces the key at index with value if index is valid else do nothing

Parameters:

index -

value -

addKey

```
public void addKey(int index,
                   E value)
```

inserts value at index if index is valid else do nothing

Parameters:

index -

value -

addFirstKey

```
public void addFirstKey(E value)
```

inserts value to start of keys list

Parameters:

value -

addLastKey

```
public void addLastKey(E value)
```

inserts value to end of keys list

Parameters:

value -

removeKey

```
public E removeKey(int index)
```

removes and returns key at index

Parameters:

index -

Returns:

removes and returns key at index

removeFirstKey

```
public E removeFirstKey()
```

removes and returns first key

Parameters:

index -

Returns:

removes and returns first key

removeLastKey

```
public E removeLastKey()
```

removes and returns the last key

Returns:

removes and returns the last key

removeChild

```
public BTreeNode<E> removeChild(int index)
```

removes and returns child at index

Parameters:

index -

Returns:

removes and returns child at index

removeFirstChild

```
public BTreeNode<E> removeFirstChild()
```

removes and return first child

Returns:

removes and return first child

removeLastChild

```
public BTreeNode<E> removeLastChild()
```

removes and return last child

Returns:

removes and return last child

findInsertPos

```
public int findInsertPos(E new_key)
```

returns the index at which new_key should be inserted in keys arraylist

Parameters:

new_key -

Returns:

returns the index at which new_key should be inserted in keys arraylist

indexOf

```
public int indexOf(E target)
```

returns the index in which target is found in this node's key arraylist or -1 if not found

Parameters:

target -

Returns:

returns the index in which target is found in this node's key arraylist or -1 if not found

indexOf

```
public int indexOf(BTreeNode<E> child)
```

returns the index in which child is found in this node's children arraylist or -1 if not found

Parameters:

child -

Returns:

returns the index in which child is found in this node's children arraylist or -1 if not found

contains

```
public boolean contains(E target)
```

returns true if this node has target in its keys

Returns:

returns true if this node has target in its keys

isOverflow

```
public boolean isOverflow()
```

returns true if there is an overflow in this node

Returns:

returns true if there is an overflow in this node

add

```
public BTreeNode<E> add(E k)
```

adds k to this node's key list in the right place returns null if there is no overflow after adding other wise:

Parameters:

k - value to add

Returns:

adds k to this node's key list in the right place returns null if there is no overflow after adding other wise:

main

```
public static void main(java.lang.String[] args)
```