# COP 5536 Spring 2021 Programming Project Report

# B+ Tree

**UFID:** 4360-0170

**Name:** Riyaz Basha Shaik

**Email:** riyaz.shaik@ufl.edu

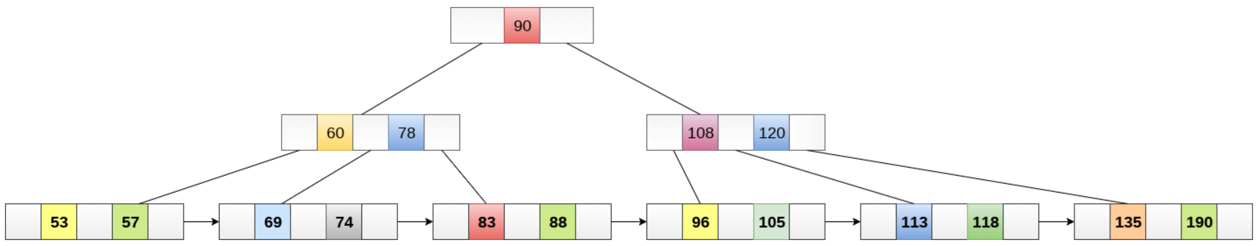
## Introduction:

A B+ tree is an N-ary tree with a variable but often large number of children per node. A B+ tree consists of a root, internal nodes and leaves. The root may be either a leaf or a node with two or more children. A B+ tree can be viewed as a B-tree in which each node contains only keys (not key–value pairs), and to which an additional level is added at the bottom with linked leaves.

The primary value of a B+ tree is in storing data for efficient retrieval in a block-oriented storage context — in particular, filesystems. This is primarily because unlike binary search trees, B+ trees have very high fanout (number of pointers to child nodes in a node, typically on the order of 100 or more), which reduces the number of I/O operations required to find an element in the tree.

Due to the fact that, size of main memory is always limited, the internal nodes (keys to access records) of the B+ tree are stored in the main memory whereas, leaf nodes are stored in the secondary memory. The internal nodes of B+ tree are often called index nodes.

A B+ tree of order 3 is shown in the following figure:



## File Structure:

**bplustree.java** is the source file which contains all the classes and methods needed to initialize a B+ Tree, perform insert, delete and search operations. This class contains main method which is the point of entry of the project. It takes the input file name from the command line argument, search for the file, opens it and reads it line by line, performing 5 kind of operations defined by the input file (initialize, insert, search by key, search between keys, delete) and writes the output of the searches to a new file, named “output_file.txt”.

The following are the classes created for this project:

- **bplustree** – This is class where B+ Tree is implemented. It supports initialize, insert, search by key, search between keys, delete operations provided through the input file.
- **Data** - Nested class to hold B+ tree node key value pair.
- **DataComparator** - Nested class to sort key value pairs in the increasing order of keys.
- **InternalNode** - Nested class to represent Internal Node of a B+ Tree. Contains Internal Node as parent and left, right sibling Internal nodes forming a doubly liked list of Internal nodes, list of keys whose values are present in the leaf nodes of its children, list of child pointers
- **LeafNode** - Nested class to represent Leaf Node of a B+ Tree. Contains Internal Node as parent and left, right sibling leaf nodes forming a doubly liked list of leaf nodes and list of key value pairs stored by the leaf node
- **OutputRecordsFormatter** - Nested class to format the output file records.

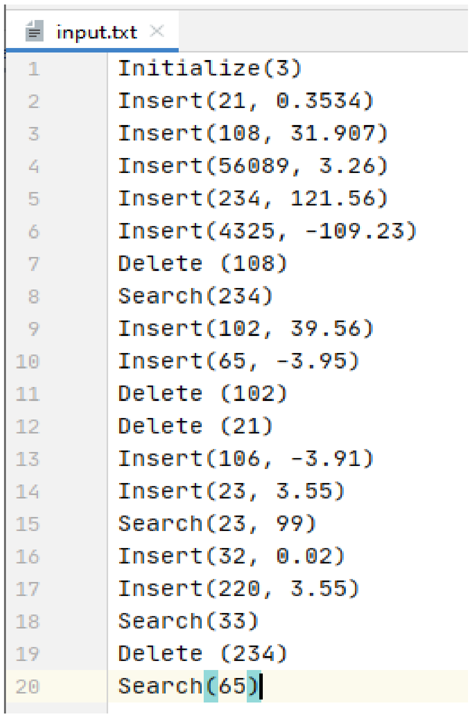
Please follow the below steps to run the project:

- **javac bplustree.java**
- **java bplustree <input_file_name>**
    - **example: java bplustree input.txt**

To run using makefile. Please follow the below steps:

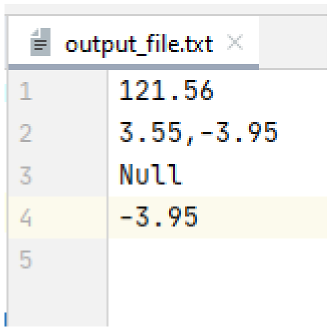
- Run **'make bplustree.class' or** just simply Run **'make'**
- Run **'make bplustree' (It assumes default input file name is input.txt)**
  or Run '**java bplustree <input_file_name>' example : java bplustree input.txt**

The following is the sample input given to the project:


input.txt

```
Initialize(3)
Insert(21, 0.3534)
Insert(108, 31.907)
Insert(56089, 3.26)
Insert(234, 121.56)
Insert(4325, -109.23)
Delete (108)
Search(234)
Insert(102, 39.56)
Insert(65, -3.95)
Delete (102)
Delete (21)
Insert(106, -3.91)
Insert(23, 3.55)
Search(23, 99)
Insert(32, 0.02)
Insert(220, 3.55)
Search(33)
Delete (234)
Search(65)
```

The following is the output generated for the above input:

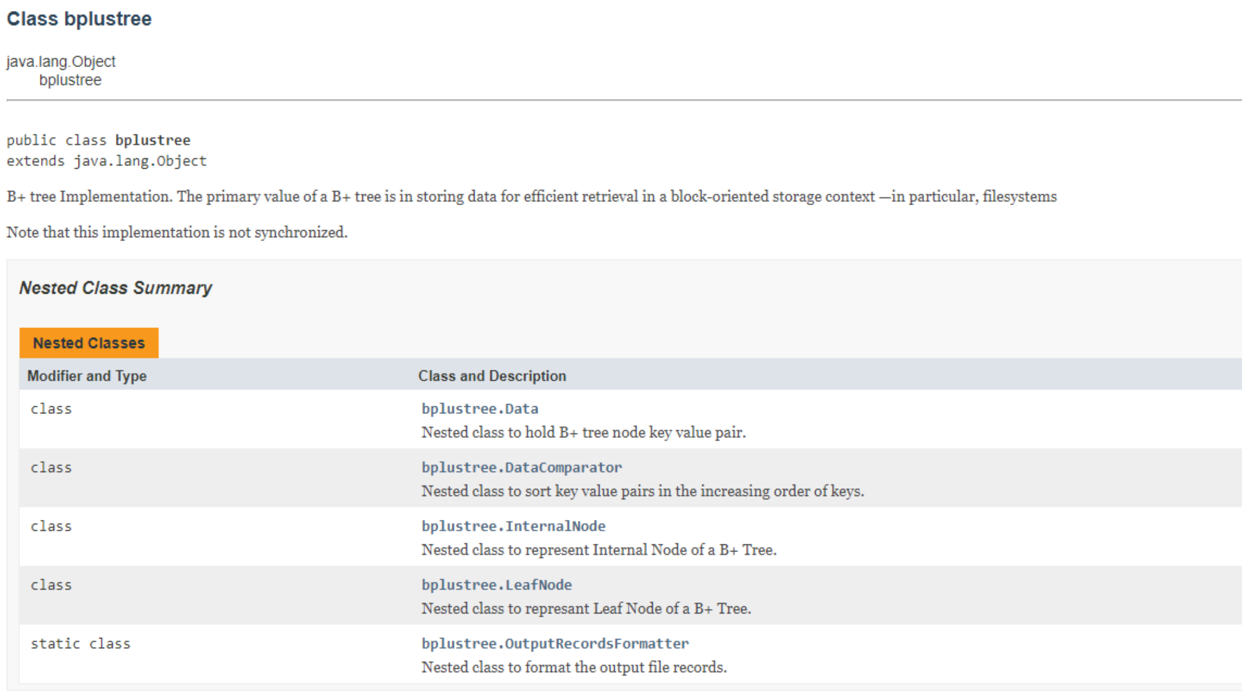


output_file.txt

```
121.56
3.55,-3.95
Null
-3.95
```

## Program Structure:

Below are the classes and methods created and their structure as part of this project:



### Class bplustree

java.lang.Object
    bplustree

```
public class bplustree
extends java.lang.Object
```

B+ tree Implementation. The primary value of a B+ tree is in storing data for efficient retrieval in a block-oriented storage context —in particular, filesystems

Note that this implementation is not synchronized.

*Nested Class Summary*

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| `class` | `bplustree.Data`<br>Nested class to hold B+ tree node key value pair. |
| `class` | `bplustree.DataComparator`<br>Nested class to sort key value pairs in the increasing order of keys. |
| `class` | `bplustree.InternalNode`<br>Nested class to represent Internal Node of a B+ Tree. |
| `class` | `bplustree.LeafNode`<br>Nested class to represant Leaf Node of a B+ Tree. |
| `static class` | `bplustree.OutputRecordsFormatter`<br>Nested class to format the output file records. |

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `int` | `degree` |
| `static java.lang.String` | `DELETE` |
| `bplustree.LeafNode` | `firstLeafNode` |
| `static java.lang.String` | `INITIALIZE` |
| `static java.lang.String` | `INSERT` |
| `int` | `internalNodeMaximumDegree` |
| `int` | `internalNodeMinimumDegree` |
| `int` | `maximumDataInLeafNode` |
| `int` | `midPointIndex` |
| `int` | `minimumDataInLeafNode` |
| `static java.lang.String` | `NULL_VALUE` |
| `bplustree.InternalNode` | `root` |
| `static java.lang.String` | `SEARCH` |
| `static java.util.logging.Logger` | `writer` |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| `bplustree(int degree)` Constructs an empty B+Tree with degree provided and initializes all the properties needed to perform operations. |

## Method Summary

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

| Modifier and Type | Method and Description |
|---|---|
| `void` | `adjustInternalNodes(bplustree.InternalNode node)` This method is used to adjust internal nodes when it becomes deficient. |
| `void` | `delete(int key)` This method is used to delete key value pair from B+ Tree whose key is provided in the arguments. |
| `int` | `getDegree()` Getter Method to get degree of the B+ Tree. |
| `bplustree.LeafNode` | `getFirstLeafNode()` Getter Method to get firstLeafNode of the B+ Tree. |
| `int` | `getInternalNodeMaximumDegree()` Getter Method to get the value of maximum number of child pointers of Internal Node of the B+ Tree. |
| `int` | `getInternalNodeMinimumDegree()` Getter Method to get the value of minimum number of child pointers of Internal Node of the B+ Tree. |
| `bplustree.LeafNode` | `getLeafNode(bplustree.InternalNode node, int key)` This method is used to get the leaf node which has the key from an internal node. |
| `int` | `getMaximumDataInLeafNode()` Getter Method to get the value of maximum number of key value pairs in Leaf Node of the B+ Tree. |
| `int` | `getMidPointIndex()` Getter Method to get the value of Index at which Keys of Internal Node and Key Value pairs in Leaf Node are to be split in the B+ Tree. |
| `int` | `getMinimumDataInLeafNode()` Getter Method to get the value of minimum number of key value pairs in Leaf Node of the B+ Tree. |
| `bplustree.InternalNode` | `getRoot()` Getter Method to get root of the B+ Tree. |
| `static void` | `initializeWriter()` This Method is used to initialize output file writer. |
| `void` | `insert(int key, double value)` This Method is used to insert a key value pair in the B+ Tree. |

| | |
|---|---|
| `static void` | `main(java.lang.String[] args)`<br>Main Method of the bplustree class. |
| `static java.lang.String` | `removeInBetweenWhiteSpaces(java.lang.String s)`<br>This Method is used to remove in between whitespaces from a string. |
| `void` | `search(int key)`<br>This method searches for a given key in the B+ Tree and prints the value of that particular key value pair to the output file. |
| `void` | `search(int lowerBound, int upperBound)`<br>This method searches for keys in the B+ Tree which are in between lowerBound and upperBound included and writes their values to output file. |
| `void` | `setDegree(int degree)`<br>Setter Method to set degree of the B+ Tree. |
| `void` | `setFirstLeafNode(bplustree.LeafNode firstLeafNode)`<br>Setter Method to set firstLeafNode of the B+ Tree. |
| `void` | `setInternalNodeMaximumDegree(int internalNodeMaximumDegree)`<br>Setter Method to set the value of maximum number of child pointers of Internal Node of the B+ Tree. |
| `void` | `setInternalNodeMinimumDegree(int internalNodeMinimumDegree)`<br>Setter Method to set the value of minimum number of child pointers of Internal Node of the B+ Tree. |
| `void` | `setMaximumDataInLeafNode(int maximumDataInLeafNode)`<br>Setter Method to set the value of maximum number of key value pairs in Leaf Node of the B+ Tree. |
| `void` | `setMidPointIndex(int midPointIndex)`<br>Setter Method to set the value of Index at which Keys of Internal Node and Key Value pairs in Leaf Node are to be split in the B+ Tree. |
| `void` | `setMinimumDataInLeafNode(int minimumDataInLeafNode)`<br>Setter Method to set the value of minimum number of key value pairs in Leaf Node of the B+ Tree. |
| `void` | `setRoot(bplustree.InternalNode root)`<br>Setter Method to set root of the B+ Tree. |
| `void` | `splitInternalNode(int midPointIndex, bplustree.InternalNode internalNode)`<br>This method is used to split the overfull node and balance the B+ tree. |

### Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

### *Field Detail*

#### writer

`public static java.util.logging.Logger writer`

#### INITIALIZE

`public static final java.lang.String INITIALIZE`

**See Also:**
Constant Field Values

#### INSERT

`public static final java.lang.String INSERT`

**See Also:**
Constant Field Values

#### DELETE

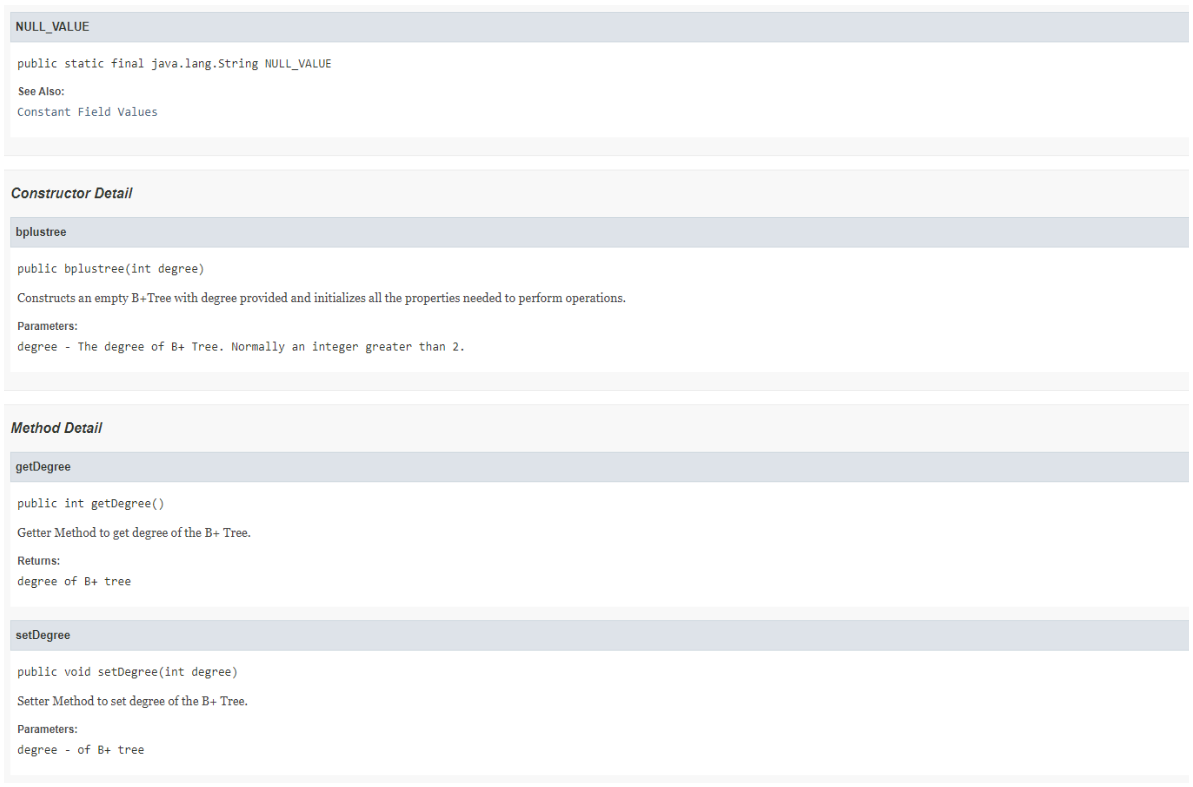
`public static final java.lang.String DELETE`

**See Also:**
Constant Field Values

#### SEARCH

`public static final java.lang.String SEARCH`

**See Also:**
Constant Field Values



**NULL_VALUE**

```
public static final java.lang.String NULL_VALUE
```

**See Also:**
Constant Field Values

### *Constructor Detail*

**bplustree**

```
public bplustree(int degree)
```

Constructs an empty B+Tree with degree provided and initializes all the properties needed to perform operations.

**Parameters:**
`degree - The degree of B+ Tree. Normally an integer greater than 2.`

### *Method Detail*

**getDegree**

```
public int getDegree()
```

Getter Method to get degree of the B+ Tree.

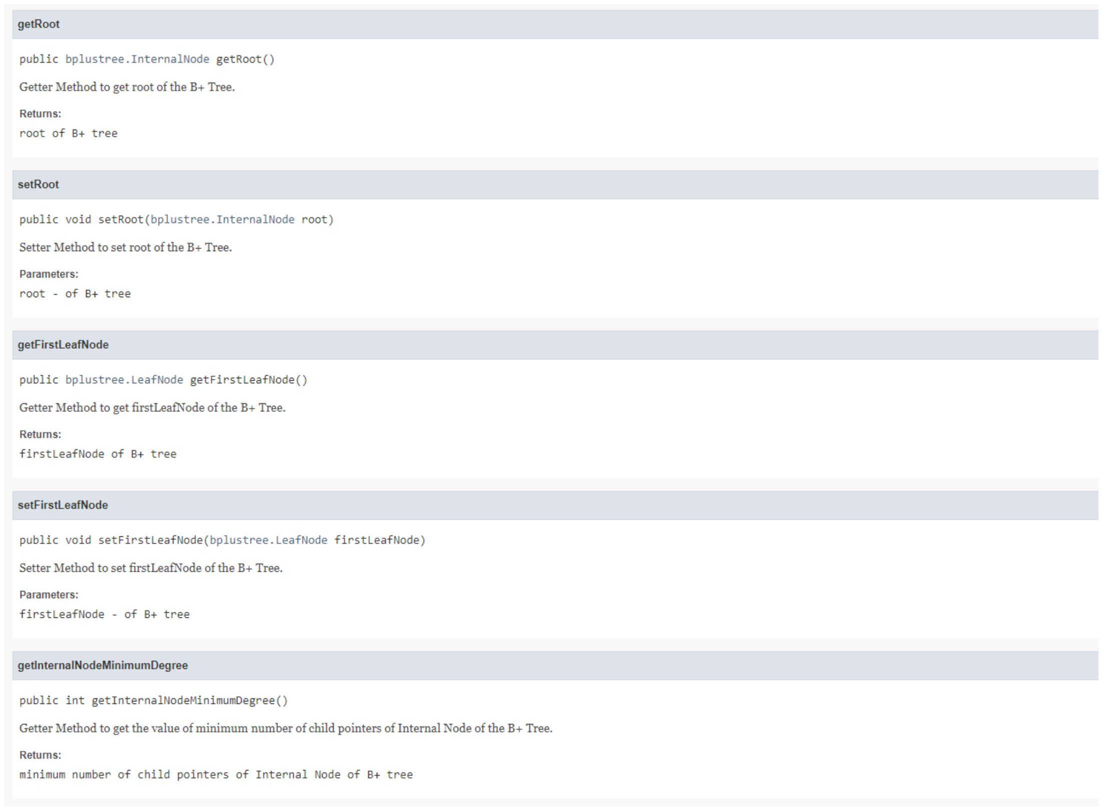
**Returns:**
`degree of B+ tree`

**setDegree**

```
public void setDegree(int degree)
```

Setter Method to set degree of the B+ Tree.

**Parameters:**
`degree - of B+ tree`



**getRoot**

```
public bplustree.InternalNode getRoot()
```

Getter Method to get root of the B+ Tree.

**Returns:**
`root of B+ tree`

**setRoot**

```
public void setRoot(bplustree.InternalNode root)
```

Setter Method to set root of the B+ Tree.

**Parameters:**
`root - of B+ tree`

**getFirstLeafNode**

```
public bplustree.LeafNode getFirstLeafNode()
```

Getter Method to get firstLeafNode of the B+ Tree.

**Returns:**
`firstLeafNode of B+ tree`

**setFirstLeafNode**

```
public void setFirstLeafNode(bplustree.LeafNode firstLeafNode)
```

Setter Method to set firstLeafNode of the B+ Tree.

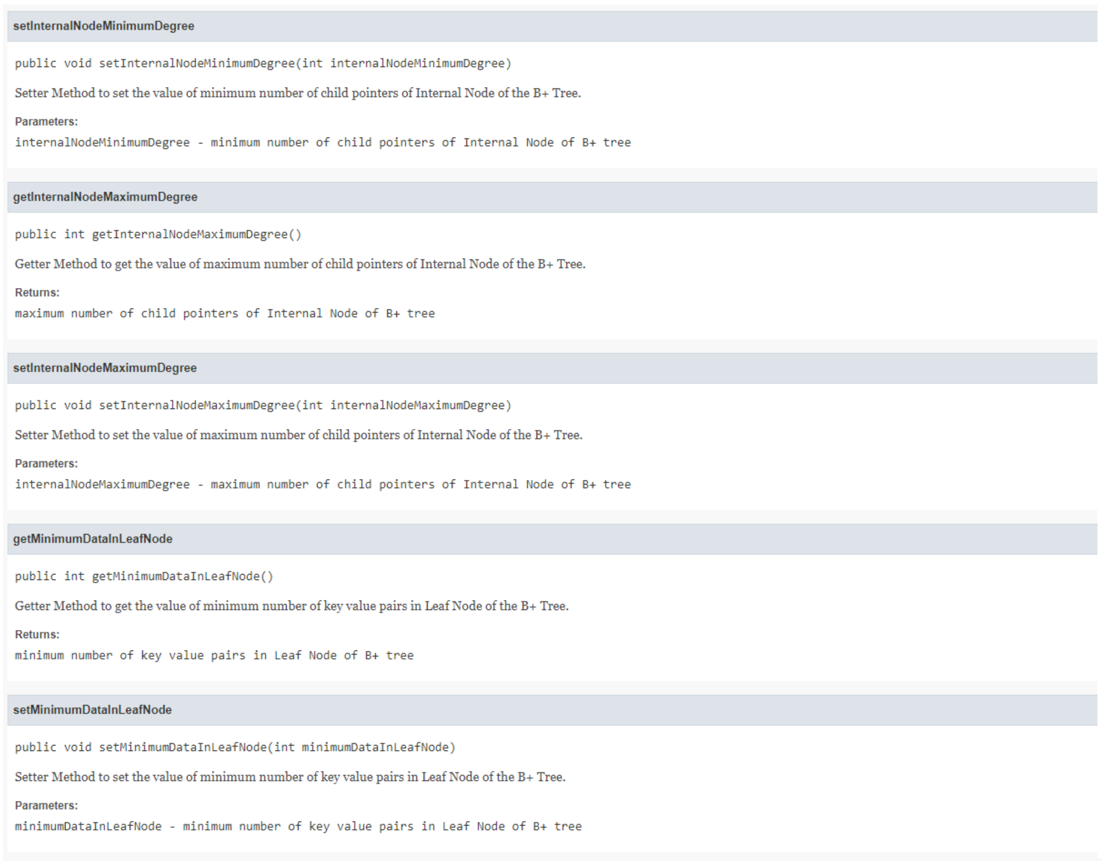
**Parameters:**
`firstLeafNode - of B+ tree`

**getInternalNodeMinimumDegree**

```
public int getInternalNodeMinimumDegree()
```

Getter Method to get the value of minimum number of child pointers of Internal Node of the B+ Tree.

**Returns:**
`minimum number of child pointers of Internal Node of B+ tree`



### setInternalNodeMinimumDegree

```
public void setInternalNodeMinimumDegree(int internalNodeMinimumDegree)
```

Setter Method to set the value of minimum number of child pointers of Internal Node of the B+ Tree.

**Parameters:**
`internalNodeMinimumDegree - minimum number of child pointers of Internal Node of B+ tree`

### getInternalNodeMaximumDegree

```
public int getInternalNodeMaximumDegree()
```

Getter Method to get the value of maximum number of child pointers of Internal Node of the B+ Tree.

**Returns:**
`maximum number of child pointers of Internal Node of B+ tree`

### setInternalNodeMaximumDegree

```
public void setInternalNodeMaximumDegree(int internalNodeMaximumDegree)
```

Setter Method to set the value of maximum number of child pointers of Internal Node of the B+ Tree.

**Parameters:**
`internalNodeMaximumDegree - maximum number of child pointers of Internal Node of B+ tree`

### getMinimumDataInLeafNode

```
public int getMinimumDataInLeafNode()
```

Getter Method to get the value of minimum number of key value pairs in Leaf Node of the B+ Tree.

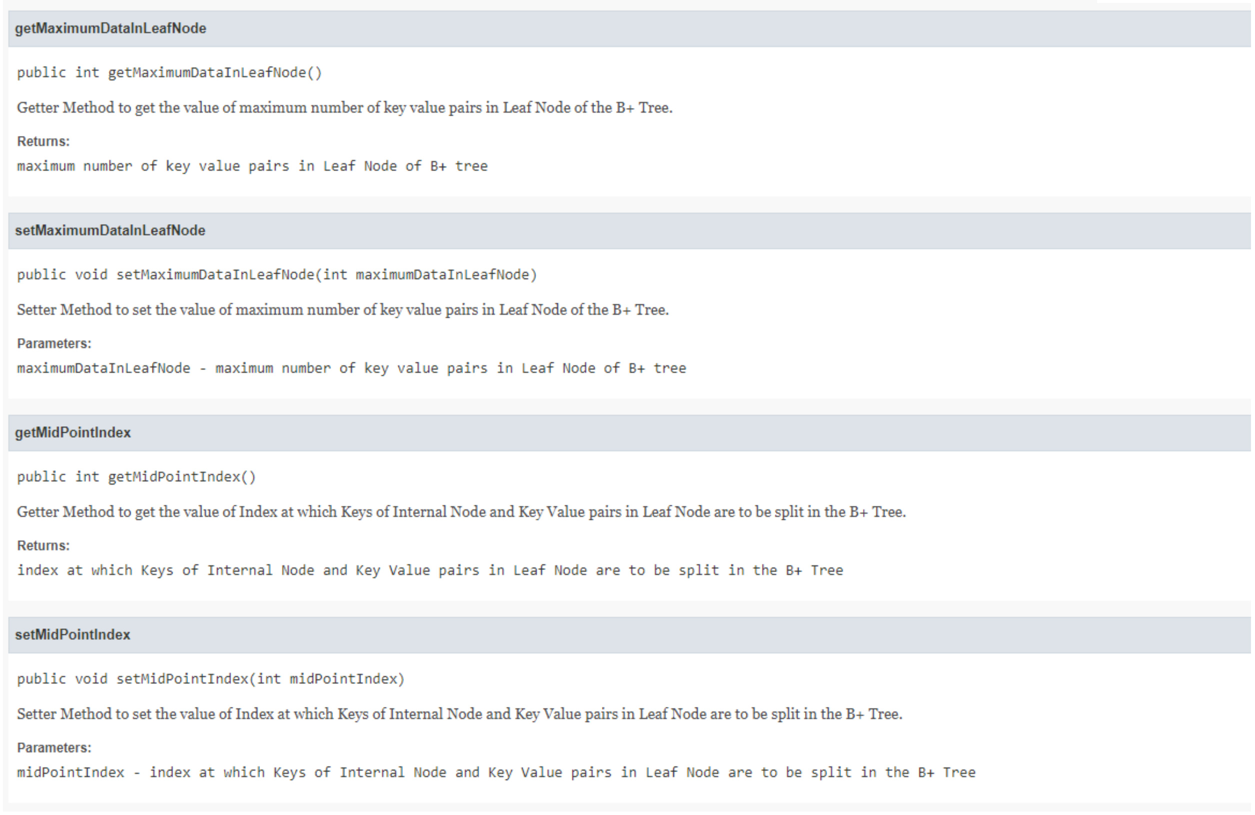
**Returns:**
`minimum number of key value pairs in Leaf Node of B+ tree`

### setMinimumDataInLeafNode

```
public void setMinimumDataInLeafNode(int minimumDataInLeafNode)
```

Setter Method to set the value of minimum number of key value pairs in Leaf Node of the B+ Tree.

**Parameters:**
`minimumDataInLeafNode - minimum number of key value pairs in Leaf Node of B+ tree`



### getMaximumDataInLeafNode

```
public int getMaximumDataInLeafNode()
```

Getter Method to get the value of maximum number of key value pairs in Leaf Node of the B+ Tree.

**Returns:**
`maximum number of key value pairs in Leaf Node of B+ tree`

### setMaximumDataInLeafNode

```
public void setMaximumDataInLeafNode(int maximumDataInLeafNode)
```

Setter Method to set the value of maximum number of key value pairs in Leaf Node of the B+ Tree.

**Parameters:**
`maximumDataInLeafNode - maximum number of key value pairs in Leaf Node of B+ tree`

### getMidPointIndex

```
public int getMidPointIndex()
```

Getter Method to get the value of Index at which Keys of Internal Node and Key Value pairs in Leaf Node are to be split in the B+ Tree.

**Returns:**
`index at which Keys of Internal Node and Key Value pairs in Leaf Node are to be split in the B+ Tree`

### setMidPointIndex

```
public void setMidPointIndex(int midPointIndex)
```

Setter Method to set the value of Index at which Keys of Internal Node and Key Value pairs in Leaf Node are to be split in the B+ Tree.

**Parameters:**
`midPointIndex - index at which Keys of Internal Node and Key Value pairs in Leaf Node are to be split in the B+ Tree`

### removeInBetweenWhiteSpaces

```
public static java.lang.String removeInBetweenWhiteSpaces(java.lang.String s)
```

This Method is used to remove in between whitespaces from a string. It accepts a string and removes any whitespaces present in it. For example `Delete (10)` becomes `Delete(10)`

**Parameters:**

`s - string whose in between whitespaces are to be removed`

**Returns:**

`string with in between whitespaces removed`

### initializeWriter

```
public static void initializeWriter()
```

This Method is used to initialize output file writer.

### insert

```
public void insert(int key,
                   double value)
```

This Method is used to insert a key value pair in the B+ Tree. It accepts a key value pair and inserts it into a leaf node and balances internal nodes from bottom to top.

**Parameters:**

`key - Key to inserted`

`value - value to inserted`

### splitInternalNode

```
public void splitInternalNode(int midPointIndex,
                              bplustree.InternalNode internalNode)
```

This method is used to split the overfull node and balance the B+ tree. This method splits keys and children lists based on midpoint index and creates a new sibling node containing half the keys and child pointers and adds the sibling node to parent

**Parameters:**

`midPointIndex - index at which keys and child pointers are to be split`

`internalNode - node to be split`

### getLeafNode

```
public bplustree.LeafNode getLeafNode(bplustree.InternalNode node,
                                      int key)
```

This method is used to get the leaf node which has the key from an internal node. It accepts Internal Node and Key and returns the Leaf Node which contains the key

**Parameters:**

`node - the Internal Node from where leaf node needs to be found`

`key - key to be found`

**Returns:**

`Leaf Node which contains the key`

### adjustInternalNodes

```
public void adjustInternalNodes(bplustree.InternalNode node)
```

This method is used to adjust internal nodes when it becomes deficient. It accepts an Internal node and checks - 1. If it can borrow a key from left or right sibling. If yes it borrows a key through parent 2. If it can merge with left or right sibling. If yes it merges with sibling and deletes in between parent key It adjust from the current node till root

**Parameters:**

`node - node to be adjusted`

### delete

```
public void delete(int key)
```

This method is used to delete key value pair from B+ Tree whose key is provided in the arguments. It accepts a key and delete corresponding key value from leaf node. After removing a key value pair if the leaf node becomes deficient. It checks if - 1. If it can borrow a key value pair from left or right sibling. If yes it borrows the pair through parent 2. If it can merge with left or right sibling. If yes it merges with sibling and deletes in between parent key After adjusting leaf node if internal node becomes deficient it adjust internal node all the way upto root

**Parameters:**

`key - key of the key value pair to be deleted`

### search

```
public void search(int key)
```

This method searches for a given key in the B+ Tree and prints the value of that particular key value pair to the output file. It accepts a key and writes value of that key if key is found else writes 'Null' to output file

**Parameters:**

`key - key to be searched`

### search

```
public void search(int lowerBound,
                   int upperBound)
```

This method searches for keys in the B+ Tree which are in between lowerBound and upperBound included and writes their values to output file. It accepts lowerBound and upperBound and writes value of those keys which fall in range of [lowerBound, upperBound] else writes 'Null' if no values are found to output file

**Parameters:**

`lowerBound` - lowerBound of the range of keys to be searched

`upperBound` - upperBound of the range of keys to be searched

### main

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

Main Method of the bplustree class. It creates B+ plus tree. It reads input file to insert values to it and delete values from it. It writes the search results to output file

**Parameters:**

`args` - Pass the name of the input file containing tree operations

## Class bplustree.Data

java.lang.Object
    bplustree.Data

**Enclosing class:**

`bplustree`

---

```
public class bplustree.Data
extends java.lang.Object
```

Nested class to hold B+ tree node key value pair.

### *Field Summary*

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `int` | `key` |
| `double` | `value` |

### *Constructor Summary*

**Constructors**

| Constructor and Description |
|---|
| `Data(int key, double value)`<br>Constructs a Data Instance with key value pairs provided in the params. |

### *Method Summary*

**All Methods** **Instance Methods** **Concrete Methods**

| Modifier and Type | Method and Description |
|---|---|
| `int` | `getKey()`<br>Getter Method to get key of the Key Value pair. |
| `double` | `getValue()`<br>Getter Method to get value of the Key Value pair. |
| `void` | `setKey(int key)`<br>Setter Method to set key of the Key Value pair. |
| `void` | `setValue(double value)`<br>Setter Method to set value of the Key Value pair. |

#### Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

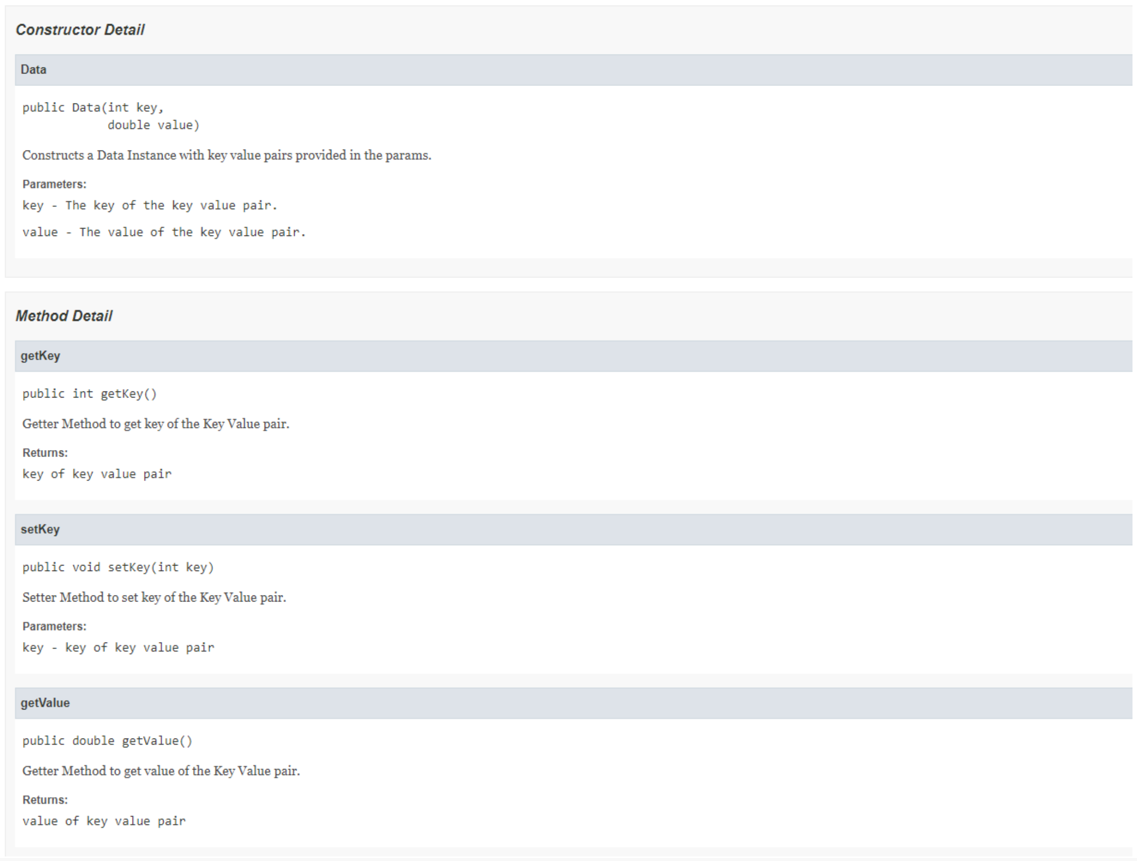
### *Field Detail*

#### key

```
public int key
```

#### value

```
public double value
```



### *Constructor Detail*

#### Data

```
public Data(int key,
            double value)
```

Constructs a Data Instance with key value pairs provided in the params.

**Parameters:**

`key - The key of the key value pair.`

`value - The value of the key value pair.`

### *Method Detail*

#### getKey

```
public int getKey()
```

Getter Method to get key of the Key Value pair.

**Returns:**

`key of key value pair`

#### setKey

```
public void setKey(int key)
```

Setter Method to set key of the Key Value pair.

**Parameters:**

`key - key of key value pair`

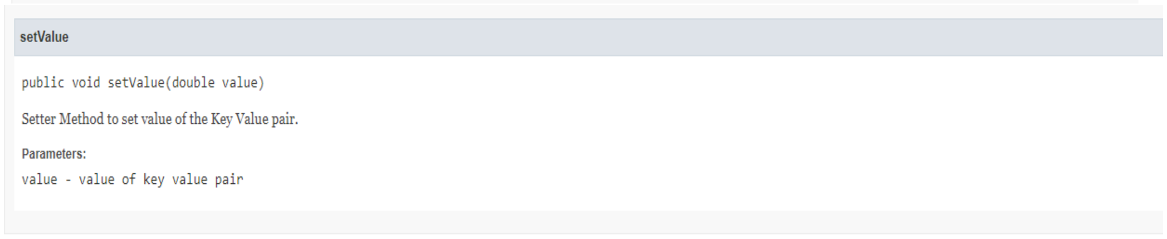
#### getValue

```
public double getValue()
```

Getter Method to get value of the Key Value pair.

**Returns:**

`value of key value pair`



#### setValue

```
public void setValue(double value)
```

Setter Method to set value of the Key Value pair.

**Parameters:**

`value - value of key value pair`

## Class bplustree.DataComparator

java.lang.Object
    bplustree.DataComparator

**All Implemented Interfaces:**

`java.util.Comparator<bplustree.Data>`

**Enclosing class:**

`bplustree`

---

```
public class bplustree.DataComparator
extends java.lang.Object
implements java.util.Comparator<bplustree.Data>
```

Nested class to sort key value pairs in the increasing order of keys.

### *Constructor Summary*

**Constructors**

| Constructor and Description |
|---|
| `DataComparator()` |

### *Method Summary*

**All Methods** **Instance Methods** **Concrete Methods**

| Modifier and Type | Method and Description |
|---|---|
| `int` | `compare(bplustree.Data d1, bplustree.Data d2)`<br>Overriding Compare method of Comparator Interface. |

#### Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

#### Methods inherited from interface java.util.Comparator

`comparing, comparing, comparingDouble, comparingInt, comparingLong, equals, naturalOrder, nullsFirst, nullsLast, reversed, reverseOrder, thenComparing, thenComparing, thenComparing, thenComparingDouble, thenComparingInt, thenComparingLong`

### *Constructor Detail*

#### DataComparator

```
public DataComparator()
```

### *Method Detail*

#### compare

```
public int compare(bplustree.Data d1,
                   bplustree.Data d2)
```

Overriding Compare method of Comparator Interface. Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first key value pair is less than, equal to, or greater than the second.

**Specified by:**

`compare in interface java.util.Comparator<bplustree.Data>`

**Parameters:**

`d1 - the first key value pair to be compared.`

`d2 - the second key value pair to be compared.`

**Returns:**

`a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.`

## Class bplustree.InternalNode

java.lang.Object
    bplustree.InternalNode

**Enclosing class:**

`bplustree`

---

```
public class bplustree.InternalNode
extends java.lang.Object
```

Nested class to represent Internal Node of a B+ Tree. Contains Internal Node as parent and left, right sibling Internal nodes forming a doubly liked list of Internal nodes, list of keys whose values are present in the leaf nodes of its children, list of child pointers

### *Field Summary*

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `int` | `degree` |
| `bplustree.InternalNode` | `leftSibling` |
| `java.util.ArrayList` | `listOfChildren` |
| `java.util.ArrayList<java.lang.Integer>` | `listOfKeys` |
| `bplustree.InternalNode` | `parentNode` |
| `bplustree.InternalNode` | `rightSibling` |

### *Constructor Summary*

**Constructors**

| Constructor and Description |
|---|
| `InternalNode(java.util.ArrayList<java.lang.Integer> keys)`<br>Constructs a o degree Internal Node Instance having keys provided in the method arguments |
| `InternalNode(java.util.ArrayList<java.lang.Integer> keys, java.util.ArrayList children)`<br>Constructs an Internal Node with degree and list of child pointers provided in the method arguments |

### *Method Summary*

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

| Modifier and Type | Method and Description |
|---|---|
| `void` | `addChildPointer(java.lang.Object node)`<br>This method is used to add a new child pointer to the exsiting list of child pointers. |
| `void` | `addChildPointer(java.lang.Object node, int index)`<br>This method is used to add a new child pointer at a specific index to the existing list of child pointers. |
| `boolean` | `checkCanBorrow(int internalNodeMinimumDegree, bplustree.InternalNode sibling)`<br>This method is used to check if the current Internal Node can borrow a Key from its sibling Node. |
| `boolean` | `checkCanMerge(int internalNodeMinimumDegree, bplustree.InternalNode sibling)`<br>This method is used to check if the current Internal Node can merge with its sibling Node. |
| `int` | `findChildIndex(java.lang.Object node)`<br>This method is used to search for a child in list of children. |
| `int` | `getDegree()`<br>Getter Method to get degree of Internal Node. |
| `bplustree.InternalNode` | `getLeftSibling()`<br>Getter Method to get left sibling of Internal Node. |
| `java.util.ArrayList` | `getListOfChildren()`<br>Getter Method to get list of child pointers of Internal Node. |
| `java.util.ArrayList<java.lang.Integer>` | `getListOfKeys()`<br>Getter Method to get list of keys of Internal Node. |
| `bplustree.InternalNode` | `getParentNode()`<br>Getter Method to get parent of Internal Node. |
| `bplustree.InternalNode` | `getRightSibling()`<br>Getter Method to get right sibling of Internal Node. |
| `void` | `setDegree(int degree)`<br>Setter Method to set degree of Internal Node. |
| `void` | `setLeftSibling(bplustree.InternalNode leftSibling)`<br>Setter Method to set left sibling of Internal Node. |

| | |
|---|---|
| `void` | `setListOfChildren(java.util.ArrayList listOfChildren)`<br>Setter Method to set list of child pointers of Internal Node. |
| `void` | `setListOfKeys(java.util.ArrayList<java.lang.Integer> listOfKeys)`<br>Setter Method to set list of keys of Internal Node. |
| `void` | `setParentNode(bplustree.InternalNode parentNode)`<br>Setter Method to set parent of Internal Node. |
| `void` | `setRightSibling(bplustree.InternalNode rightSibling)`<br>Setter Method to set right sibling of Internal Node. |
| `void` | `sortKeys()`<br>This method is used to sort keys of the internal node in the increasing order of keys. |
| `java.util.ArrayList` | `splitChildPointers(int midPointIndex)`<br>This method is used to split the list of child pointers on a midpoint index into 2 separate lists of child pointers. |
| `java.util.ArrayList<java.lang.Integer>` | `splitKeys(int midPointIndex)`<br>This method is used to split the list of keys on a midpoint index into 2 separate lists of keys. |

**Methods inherited from class java.lang.Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

## *Constructor Detail*

### InternalNode

```
public InternalNode(java.util.ArrayList<java.lang.Integer> keys)
```

Constructs a 0 degree Internal Node Instance having keys provided in the method arguments

**Parameters:**

`keys - The list of keys`

### InternalNode

```
public InternalNode(java.util.ArrayList<java.lang.Integer> keys,
                    java.util.ArrayList children)
```

Constructs an Internal Node with degree and list of child pointers provided in the method arguments

**Parameters:**

`keys - The list of keys`

`children - The list of child pointers`

## *Method Detail*

### getDegree

```
public int getDegree()
```

Getter Method to get degree of Internal Node.

**Returns:**

`degree of Internal Node`

### setDegree

```
public void setDegree(int degree)
```

Setter Method to set degree of Internal Node.

**Parameters:**

`degree - degree of Internal Node`

### getLeftSibling

```
public bplustree.InternalNode getLeftSibling()
```

Getter Method to get left sibling of Internal Node.

**Returns:**

`left sibling of Internal Node`

### setLeftSibling

```
public void setLeftSibling(bplustree.InternalNode leftSibling)
```

Setter Method to set left sibling of Internal Node.

**Parameters:**

`leftSibling - left sibling of Internal Node`

### getRightSibling

```
public bplustree.InternalNode getRightSibling()
```

Getter Method to get right sibling of Internal Node.

**Returns:**

`right sibling of Internal Node`

### setRightSibling

```
public void setRightSibling(bplustree.InternalNode rightSibling)
```

Setter Method to set right sibling of Internal Node.

**Parameters:**

`rightSibling - right sibling of Internal Node`

### getParentNode

```
public bplustree.InternalNode getParentNode()
```

Getter Method to get parent of Internal Node.

**Returns:**

`parent of Internal Node`

### setParentNode

```
public void setParentNode(bplustree.InternalNode parentNode)
```

Setter Method to set parent of Internal Node.

**Parameters:**

`parentNode - parent of Internal Node`

### getListOfKeys

```
public java.util.ArrayList<java.lang.Integer> getListOfKeys()
```

Getter Method to get list of keys of Internal Node.

**Returns:**

`list of keys of Internal Node`

### setListOfKeys

```
public void setListOfKeys(java.util.ArrayList<java.lang.Integer> listOfKeys)
```

Setter Method to set list of keys of Internal Node.

**Parameters:**

`listOfKeys - list of keys of Internal Node`

### getListOfChildren

```
public java.util.ArrayList getListOfChildren()
```

Getter Method to get list of child pointers of Internal Node.

**Returns:**

`list of child pointers of Internal Node`

### setListOfChildren

```
public void setListOfChildren(java.util.ArrayList listOfChildren)
```

Setter Method to set list of child pointers of Internal Node.

**Parameters:**

`listOfChildren - list of child pointers of Internal Node`

### sortKeys

```
public void sortKeys()
```

This method is used to sort keys of the internal node in the increasing order of keys.

### addChildPointer

```
public void addChildPointer(java.lang.Object node)
```

This method is used to add a new child pointer to the exsiting list of child pointers.

**Parameters:**

`node - The child to be added to the existing list of children`

### addChildPointer

```
public void addChildPointer(java.lang.Object node,
                            int index)
```

This method is used to add a new child pointer at a specific index to the existing list of child pointers. The new child is inserted at position index and children previously at position index and above are pushed right.

**Parameters:**

`node - The child to be added to the existing list of children`

`index - The index at which the child needs to be added`

### findChildIndex

```
public int findChildIndex(java.lang.Object node)
```

This method is used to search for a child in list of children. It returns the index of child if the child is found else it returns -1.

**Parameters:**

`node - The child to be found.`

**Returns:**

`index index of the key if key value pair is present, -1 if key is not present`

### splitKeys

```
public java.util.ArrayList<java.lang.Integer> splitKeys(int midPointIndex)
```

This method is used to split the list of keys on a midpoint index into 2 separate lists of keys.

It returns back the list containing the second half of keys and removes those elements from the original list.

**Parameters:**

`midPointIndex - The index on which the list of keys are to be split`

**Returns:**

`The list containing the second half of keys`

### splitChildPointers

```
public java.util.ArrayList splitChildPointers(int midPointIndex)
```

This method is used to split the list of child pointers on a midpoint index into 2 separate lists of child pointers.

It returns back the list containing the second half of child pointers and removes those elements from the original list.

**Parameters:**

`midPointIndex - The index on which the list of keys are to be split`

**Returns:**

`The list containing the second half of child pointers`

### checkCanBorrow

```
public boolean checkCanBorrow(int internalNodeMinimumDegree,
                              bplustree.InternalNode sibling)
```

This method is used to check if the current Internal Node can borrow a Key from its sibling Node.

**Parameters:**

`internalNodeMinimumDegree` - The minimum number of children that internal node can have

`sibling` - The sibling of the current internal node

**Returns:**

`true If Internal Node can borrow a key from its sibling, false If Internal Node cannot borrow a key from its sibling`

### checkCanMerge

```
public boolean checkCanMerge(int internalNodeMinimumDegree,
                             bplustree.InternalNode sibling)
```

This method is used to check if the current Internal Node can merge with its sibling Node.

**Parameters:**

`internalNodeMinimumDegree` - The minimum number of key value pairs that leaf node can hold

`sibling` - The sibling of the current internal node

**Returns:**

`true If Internal Node can merge with its sibling, false If Internal Node cannot merge with its sibling`

## Class bplustree.LeafNode

java.lang.Object
    bplustree.LeafNode

**Enclosing class:**

`bplustree`

```
public class bplustree.LeafNode
extends java.lang.Object
```

Nested class to represant Leaf Node of a B+ Tree. Contains Internal Node as parent and left, right sibling leaf nodes forming a doubly liked list of leaf nodes and list of key value pairs stored by the leaf node

### *Field Summary*

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `bplustree.LeafNode` | `leftSibling` |
| `java.util.ArrayList<bplustree.Data>` | `listOfData` |
| `int` | `numberOfPairs` |
| `bplustree.InternalNode` | `parent` |
| `bplustree.LeafNode` | `rightSibling` |

### *Constructor Summary*

**Constructors**

| Constructor and Description |
|---|
| `LeafNode()` Constructs a Leaf Node Instance having no key value pairs and no left and right sibling |
| `LeafNode(java.util.ArrayList<bplustree.Data> dataList, bplustree.InternalNode parent)` Constructs a Leaf Node Instance having key value pairs provided in the method arguments and with parent value provided in the method arguments |

### *Method Summary*

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

| Modifier and Type | Method and Description |
|---|---|
| `boolean` | `checkCanBorrow(int minimumDataInLeafNode, bplustree.LeafNode sibling)` This method is used to check if the current Leaf Node can borrow a Key Value pair from its sibling Node. |
| `boolean` | `checkCanMerge(int minimumDataInLeafNode, bplustree.LeafNode sibling)` This method is used to check if the current Leaf Node can merge with its sibling Node. |
| `int` | `findIndexOfKeyInData(int key)` This method is used to find the index of a key in the list of key value pairs of current Leaf Node. |

| | |
|---|---|
| `bplustree.LeafNode` | `getLeftSibling()`<br>Getter Method to get left sibling in DLL of Leaf Node. |
| `java.util.ArrayList<bplustree.Data>` | `getListOfData()`<br>Getter Method to get list of key value pairs of Leaf Node. |
| `int` | `getNumberOfPairs()`<br>Getter Method to get number of Key Value pairs in Leaf Node. |
| `bplustree.InternalNode` | `getParent()`<br>Getter Method to get parent of Leaf Node. |
| `bplustree.LeafNode` | `getRightSibling()`<br>Getter Method to get right sibling in DLL of Leaf Node. |
| `boolean` | `insertData(int leafNodeMaximumPairs, bplustree.Data data)`<br>This method adds new key value pair (Data) to Leaf Node. |
| `void` | `setLeftSibling(bplustree.LeafNode leftSibling)`<br>Setter Method to set left sibling in DLL of Leaf Node. |
| `void` | `setListOfData(java.util.ArrayList<bplustree.Data> listOfData)`<br>Setter Method to set list of key value pairs of Leaf Node. |
| `void` | `setNumberOfPairs(int numberOfPairs)`<br>Setter Method to set number of Key Value pairs in Leaf Node. |
| `void` | `setParent(bplustree.InternalNode parent)`<br>Setter Method to set parent of Leaf Node. |
| `void` | `setRightSibling(bplustree.LeafNode rightSibling)`<br>Setter Method to set right sibling in DLL of Leaf Node. |
| `void` | `sortData()`<br>This method is used to sort key value pairs in increasing order of keys. |
| `java.util.ArrayList<bplustree.Data>` | `splitDataList(int midPointIndex)`<br>This method is used to split the list of key value pairs on a midpoint index into 2 seperate lists of key value pairs. |

**Methods inherited from class java.lang.Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

### *Constructor Detail*

#### LeafNode

```
public LeafNode()
```

Constructs a Leaf Node Instance having no key value pairs and no left and right sibling

#### LeafNode

```
public LeafNode(java.util.ArrayList<bplustree.Data> dataList,
                bplustree.InternalNode parent)
```

Constructs a Leaf Node Instance having key value pairs provided in the method arguments and with parent value provided in the method arguments

**Parameters:**

`dataList` - The list of key value pairs

`parent` - The parent of the Leaf Node

### *Method Detail*

#### getNumberOfPairs

```
public int getNumberOfPairs()
```

Getter Method to get number of Key Value pairs in Leaf Node.

**Returns:**

number of Key Value pairs in Leaf Node

#### setNumberOfPairs

```
public void setNumberOfPairs(int numberOfPairs)
```

Setter Method to set number of Key Value pairs in Leaf Node.

**Parameters:**

`numberOfPairs` - number of Key Value pairs in Leaf Node

### getLeftSibling

```
public bplustree.LeafNode getLeftSibling()
```

Getter Method to get left sibling in DLL of Leaf Node.

**Returns:**
`left sibling in DLL of Leaf Node`

### setLeftSibling

```
public void setLeftSibling(bplustree.LeafNode leftSibling)
```

Setter Method to set left sibling in DLL of Leaf Node.

**Parameters:**
`leftSibling - left sibling in DLL of Leaf Node`

### getRightSibling

```
public bplustree.LeafNode getRightSibling()
```

Getter Method to get right sibling in DLL of Leaf Node.

**Returns:**
`right sibling in DLL of Leaf Node`

### setRightSibling

```
public void setRightSibling(bplustree.LeafNode rightSibling)
```

Setter Method to set right sibling in DLL of Leaf Node.

**Parameters:**
`rightSibling - right sibling in DLL of Leaf Node`

### getListOfData

```
public java.util.ArrayList<bplustree.Data> getListOfData()
```

Getter Method to get list of key value pairs of Leaf Node.

**Returns:**
`list of key value pairs of Leaf Node`

### setListOfData

```
public void setListOfData(java.util.ArrayList<bplustree.Data> listOfData)
```

Setter Method to set list of key value pairs of Leaf Node.

**Parameters:**
`listOfData - list of key value pairs of Leaf Node`

### getParent

```
public bplustree.InternalNode getParent()
```

Getter Method to get parent of Leaf Node.

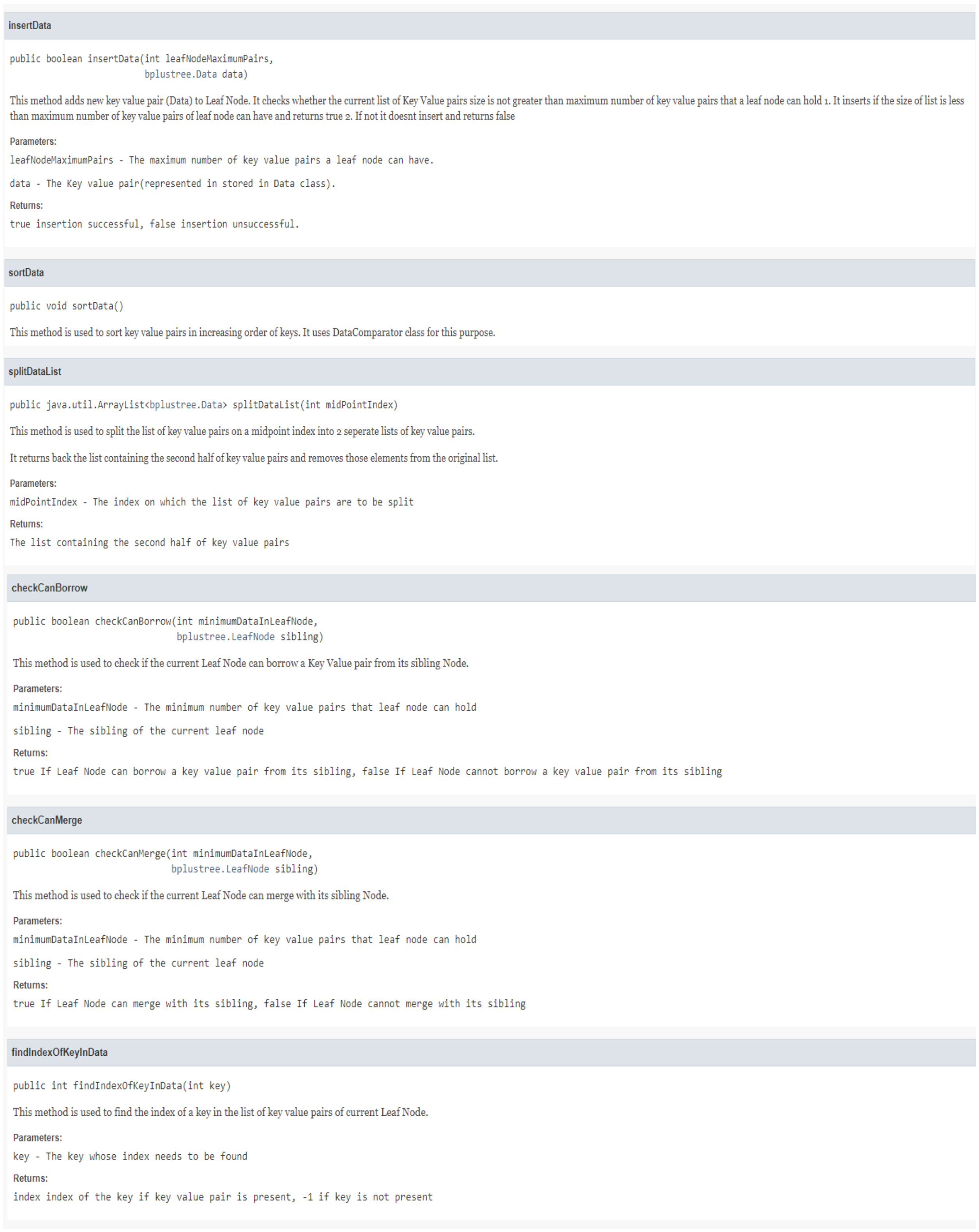
**Returns:**
`parent (Internal Node) of Leaf Node`

### setParent

```
public void setParent(bplustree.InternalNode parent)
```

Setter Method to set parent of Leaf Node.

**Parameters:**
`parent - parent (Internal Node) of Leaf Node`



**insertData**

```
public boolean insertData(int leafNodeMaximumPairs,
                          bplustree.Data data)
```

This method adds new key value pair (Data) to Leaf Node. It checks whether the current list of Key Value pairs size is not greater than maximum number of key value pairs that a leaf node can hold 1. It inserts if the size of list is less than maximum number of key value pairs of leaf node can have and returns true 2. If not it doesnt insert and returns false

**Parameters:**

`leafNodeMaximumPairs` - The maximum number of key value pairs a leaf node can have.

`data` - The Key value pair(represented in stored in Data class).

**Returns:**

true insertion successful, false insertion unsuccessful.

**sortData**

```
public void sortData()
```

This method is used to sort key value pairs in increasing order of keys. It uses DataComparator class for this purpose.

**splitDataList**

```
public java.util.ArrayList<bplustree.Data> splitDataList(int midPointIndex)
```

This method is used to split the list of key value pairs on a midpoint index into 2 seperate lists of key value pairs.

It returns back the list containing the second half of key value pairs and removes those elements from the original list.

**Parameters:**

`midPointIndex` - The index on which the list of key value pairs are to be split

**Returns:**

The list containing the second half of key value pairs

**checkCanBorrow**

```
public boolean checkCanBorrow(int minimumDataInLeafNode,
                              bplustree.LeafNode sibling)
```

This method is used to check if the current Leaf Node can borrow a Key Value pair from its sibling Node.

**Parameters:**

`minimumDataInLeafNode` - The minimum number of key value pairs that leaf node can hold

`sibling` - The sibling of the current leaf node

**Returns:**

true If Leaf Node can borrow a key value pair from its sibling, false If Leaf Node cannot borrow a key value pair from its sibling

**checkCanMerge**

```
public boolean checkCanMerge(int minimumDataInLeafNode,
                             bplustree.LeafNode sibling)
```

This method is used to check if the current Leaf Node can merge with its sibling Node.

**Parameters:**

`minimumDataInLeafNode` - The minimum number of key value pairs that leaf node can hold

`sibling` - The sibling of the current leaf node

**Returns:**

true If Leaf Node can merge with its sibling, false If Leaf Node cannot merge with its sibling

**findIndexOfKeyInData**

```
public int findIndexOfKeyInData(int key)
```

This method is used to find the index of a key in the list of key value pairs of current Leaf Node.

**Parameters:**

`key` - The key whose index needs to be found

**Returns:**

index index of the key if key value pair is present, -1 if key is not present

## Class bplustree.OutputRecordsFormatter

java.lang.Object
    java.util.logging.Formatter
        bplustree.OutputRecordsFormatter

**Enclosing class:**

`bplustree`

---

```
public static class bplustree.OutputRecordsFormatter
extends java.util.logging.Formatter
```

Nested class to format the output file records.

### *Constructor Summary*

**Constructors**

| Constructor and Description |
|---|
| `OutputRecordsFormatter()` |

### *Method Summary*

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

| Modifier and Type | Method and Description |
|---|---|
| `java.lang.String` | `format(java.util.logging.LogRecord record)`<br>This Method is used to format the output file records. |
| `java.lang.String` | `formattedMessage(java.lang.String message)`<br>This Method is used to format the message. |

#### Methods inherited from class java.util.logging.Formatter

`formatMessage, getHead, getTail`

#### Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

### *Constructor Detail*

#### OutputRecordsFormatter

```
public OutputRecordsFormatter()
```

### *Method Detail*

#### format

```
public java.lang.String format(java.util.logging.LogRecord record)
```

This Method is used to format the output file records. It accepts a log record and returns the formatted output record

**Specified by:**

`format in class java.util.logging.Formatter`

**Parameters:**

`record - output file record`

**Returns:**

`formatted output file record`

#### formattedMessage

```
public java.lang.String formattedMessage(java.lang.String message)
```

This Method is used to format the message. It accepts a message and formats it

**Parameters:**

`message - message to be formatted`

**Returns:**

`formatted message`