13. Trees :: Binary Trees :: Java Implementation :: *BinaryTree<E> iterator*()

In data structures, an **iterator** is an object which references an element of the structure and permits us to move around among the elements of a data structure as we perform operations on those elements.

The *BinaryTree<E>* class encapsulates a static nested class named *Iterator<E>* which provides iteration methods. We will discuss the *Iterator<E>* class in more detail later, but for now, the *BinaryTree* class *iterator*() method can be called to create an *Iterator<E>* object:

```
public class BinaryTree<E> {
  ...
  // Creates a new Iterator over this BinaryTree. The current node of the returned
  // Iterator will be the root node of this BinaryTree.
  public Iterator<E> iterator() {
    return new Iterator(this);
  }
}
```

To create an iterator:

```
  BinaryTree<Integer> tree = new BinaryTree<>(1);
  BinaryTree.Iterator<Integer> it = tree.iterator();
```

13. Trees :: Binary Trees :: Java Implementation :: *BinaryTree.Iterator<E>* Methods

`+addLeft(pData: E): void`

Creates a new *Node* containing *pData* to be the left child of this *Iterator*'s current *Node*. If the current *Node* already has a left child, the subtree rooted at the left child is pruned.

`+addRight(pData: E): void`

Creates a new *Node* containing *pData* to be the right child of this *Iterator*'s current *Node*. If the current *Node* already has a right child, the subtree rooted at the right child is pruned.

For example:

```
BinaryTree<Integer> tree = new BinaryTree<>(1);
BinaryTree.Iterator<Integer> it = tree.iterator();
it.addLeft(2); it.addRight(3);
```

13. Trees :: Binary Trees :: Java Implementation :: *BinaryTree.Iterator<E>* Methods

`+moveLeft(): void`

Moves this *Iterator* to the left child of this *Iterator*'s current *Node*.

`+moveRight(): void`

Moves this *Iterator* to the right child of this *Iterator*'s current *Node*.

`+moveToRoot(): void`

Moves this *Iterator* to the root *Node* of the tree over which this *Iterator* iterates.

`+moveUp(): void`

Moves this *Iterator* up to the parent *Node* of this *Iterator*'s current *Node*.

For example:

```
it.moveLeft(); it.addLeft(4); it.addRight(5);
it.moveUp(); it.moveRight(); it.addLeft(6); it.addRight(7);
it.moveToRoot();
```