16. Trees :: Binary Trees :: Java Implementation :: *BinaryTree<E>* Traversals

The *BinaryTree*<E> class implements three methods which are used during traversals.

`+traverse(pWhich: int, pVisitor: BinaryTreeVisitor): void`

Performs a traversal on this *BinaryTree*. *pWhich* is one of the static constants declared in *BinaryTree<E>* INORDER, LEVEL_ORDER, POSTORDER, or PREORDER. *pVisitor* is an object which implements the *BinaryTreeVisitor<E>* interface:

```
public interface BinaryTreeVisitor<E> {
  void visit(E pData);
}
```

visit() is called once per *Node* during the traversal and it may do anything with *pData* as it sees fit. The implementation of *traverse*(**int**, *BinaryTreeVisitor<E>*) depends on the other two *traverse*() methods that we will discuss in the next section:

```
// A level order traversal is performed in a completely different manner than
// inorder, preorder, and postorder traversals so we call a separate method to
// perform aa level order traversal. Otherwise, we call the other traverse()
// method to perform a traversal starting at the root node of this BinaryTree.
public void traverse(int pWhich, BinaryTreeVisitor<E> pVisitor) {
  if (pWhich == LEVEL_ORDER) traverseLevelOrder(getRoot(), pVisitor);
  traverse(pWhich, getRoot(), pVisitor);
}
```

16. Trees :: Binary Trees :: Java Implementation :: *BinaryTree<E>* Traversals

For example:

```
public class Main implements BinaryTreeVisitor<Integer> {
  public static void main(String[] pArgs) { new Main().run(); }
  private void run() {
    BinaryTree<Integer> tree = new BinaryTree<>(1);
    BinaryTree.Iterator it = tree.iterator();
    it.addLeft(2); it.addRight(3);
    it.moveLeft(); it.addLeft(4); it.addRight(5);
    it.moveUp(); it.moveRight(); it.addLeft(6); it.addRight(7);
    tree.traverse(BinaryTree.INORDER, this); System.out.println();
    tree.traverse(BinaryTree.LEVEL_ORDER, this); System.out.println();
    tree.traverse(BinaryTree.POSTORDER, this); System.out.println();
    tree.traverse(BinaryTree.PREORDER, this); System.out.println();
    it.moveToRoot(); it.moveRight();
    it.traverse(BinaryTree.INORDER, this); System.out.println();
    it.traverse(BinaryTree.LEVEL_ORDER, this); System.out.println();
    it.traverse(BinaryTree.POSTORDER, this); System.out.println();
    it.traverse(BinaryTree.PREORDER, this); System.out.println();
  }
  public visit (Integer pData) { System.out.print(pData + " "); }
}
```

16. Trees :: Binary Trees :: Java Implementation :: *BinaryTree<E>* Traversals

Output

```
4 2 5 1 6 3 7
1 2 3 4 5 6 7
4 5 2 6 7 3 1
1 2 4 5 3 6 7
6 3 7
3 6 7
6 7 3
3 6 7
```