

DELETION FROM A BINARY TREE

```
public void delete(Comparable target)
// post: deletes a node with data equal to target, if present,
//         preserving binary search tree property
{
    myRoot = deleteHelper(myRoot, target);
}

private TreeNode deleteHelper(TreeNode node, Comparable target)
// pre : node points to a non-empty binary search tree
// post: deletes a node with data equal to target, if present,
//         preserving binary search tree property
{
    if (node == null)
        throw new NoSuchElementException();

    else if (target.compareTo(node.getValue()) == 0)
    {
        return deleteTargetNode(node);
    }
    else if (target.compareTo(node.getValue()) < 0)
    {
        node.setLeft(deleteHelper(node.getLeft(), target));
        return node;
    }
    else //target.compareTo(root.getValue()) > 0
    {
        node.setRight(deleteHelper(node.getRight(), target));
        return node;
    }
}

private TreeNode deleteTargetNode(TreeNode target)
// pre : target points to node to be deleted
// post: target node is deleted preserving binary search tree property
{
    if (target.getRight() == null)
    {
        return target.getLeft();
    }
    else if (target.getLeft() == null)
    {
        return target.getRight();
    }
    else if (target.getLeft().getRight() == null)
    {
        target.setValue(target.getLeft().getValue());
        target.setLeft(target.getLeft().getLeft());
        return target;
    }
    else // left child has right child
    {
        TreeNode marker = target.getLeft();
        while (marker.getRight().getRight() != null)
            marker = marker.getRight();

        target.setValue(marker.getRight().getValue());
        marker.setRight(marker.getRight().getLeft());
    }
}
```

```

        return target;
    }
}

// ----- testDelete method - add to BSTree.java

public void testDelete(BinarySearchTree temp)
{
    int idToDelete;
    boolean success;

    System.out.println("Testing delete algorithm\n");
    System.out.print("Enter Id value to delete (-1 to quit) --> ");
    idToDelete = console.readInt();

    while (idToDelete >= 0)
    {
        Item dNode = new Item(idToDelete, 0);

        if (temp.find(dNode) == null)
            System.out.println("Id# " + idToDelete + " No such part in stock");
        else
        {
            temp.delete(dNode);
            System.out.println("      Id #" + idToDelete + " was deleted");
        }
        System.out.println();
        System.out.print("Enter Id value to delete (-1 to quit) --> ");

        idToDelete = console.readInt();
    }
}

```