Binary Search Tree Implementation

BinarySearchTree

```
public class BinarySearchTree <T  extends Comparable<? super T>> {

    private class Node {
        private Node left;
        private T data;
        private Node right;
        private int count;

        private Node(Node l, T d, Node r) {
            left = l;
            data = d;
            right = r;
            count = 1;
        }
    }

    private Node root;
    public BinarySearchTree() {
        root = null;
    }
```

BinarySearchTree

```
public void insert(T d) {
	root = insert(root, d);
}

private Node insert(Node r, T d) {
	if (r == null) return new Node(null, d, null);
	if (r.data.compareTo(d) > 0) r.left = insert(r.left, d);
	else if (r.data.compareTo(d) < 0) r.right = insert(r.right, d);
	else r.count++;
	return r;
}
```

BinarySearchTree

```
public int getCount(T d) {
	return getCount(root, d);
}

private int getCount(Node r, T d) {
	if (r == null) return 0;
	if (r.data.compareTo(d) > 0) return getCount(r.left, d);
	if (r.data.compareTo(d) < 0) return getCount(r.right, d);
	return r.count;
}
```

BinarySearchTree

```
public T getData(T d) {
	return getData(root, d);
}

private T getData(Node r, T d) {
	if (r == null) return null;
	if (r.data.compareTo(d) > 0) return getData(r.left, d);
	if (r.data.compareTo(d) < 0) return getData(r.right, d);
	return r.data;
}
```

BinarySearchTree

```
public void remove(T d) {
      root = remove(root, d);
}

private Node remove(Node r, T d) {
      if (r == null) return null;
      Node retVal = r ;
      if (r.data.compareTo(d) == 0) {
            r.count--;
            if (r.count == 0) {
                  if (r.left == null) retVal = r.right;
                  else if (r.right == null) retVal = r.left;
                  else r.left = replace (r.left, r);
            }

      }
      else if (r.data.compareTo(d) > 0) r.left = remove(r.left, d);
      else r.right = remove(r.right, d);
      return retVal;
}
```

BinarySearchTree

```
private Node replace(Node r, Node repHere) {
      if (r.right != null) {
            r.right = replace(r.right, repHere);
           return r;
      }
      else {
           repHere.data = r.data;
           repHere.count = r.count;
           return r.left;
      }
}
```