```java
// BinarySearchTree.java          by Dale/Joyce/Weems                Chapter 8

package bst_Package;

import org.w3c.dom.Node;

public class BinarySearchTree<T extends Comparable<T>>
             implements BSTInterface<T>
{
  protected BSTNode<T> root;      // reference to the root of this BST

  boolean found;   // used by remove

  // for traversals
  protected LinkedUnbndQueue<T> inOrderQueue;    // queue of info
  protected LinkedUnbndQueue<T> preOrderQueue;   // queue of info
  protected LinkedUnbndQueue<T> postOrderQueue;  // queue of info

  public BinarySearchTree()
  // Creates an empty BST object.
  {
    root = null;
  }

  // Recursively count each leaf node on BST
  // Returns count of leaves
  private int recLeafCount(BSTNode<T> node) {
      int count = 0;
      if (node == null)
            count += 0;
        else if(node.getLeft() == null && node.getRight() == null)
            count = 1;
        else
            count += recLeafCount(node.getLeft()) + recLeafCount(node.getRight());

            return count;
    }

  // Count the leaf nodes on the Binary Search Tree
    public int leafCount() {
        return recLeafCount(root);
    }

    // Count the single parents on the Binary Search Tree
    public int singleParentCount() {
        return recSingleParentCount(root);
    }

    // Recursively count all the single the parents on the Binary Search tree
    // Returns count of single parents
    private int recSingleParentCount(BSTNode<T> node) {
        if (node == null)
            return 0;
        else if ((node.getLeft() == null && node.getRight() != null)||
                (node.getLeft() != null && node.getRight() == null))
            return 1;
        else
```

```java
            return recSingleParentCount(node.left) + recSingleParentCount(node.right);
    }

  public boolean isEmpty()

  private int recSize(BSTNode<T> tree)

  public int size()

  public int size2()

  private boolean recContains(T element, BSTNode<T> tree)

  public boolean contains (T element)

  private T recGet(T element, BSTNode<T> tree)

  public T get(T element)

  private BSTNode<T> recAdd(T element, BSTNode<T> tree)

  public void add (T element)

  private T getPredecessor(BSTNode<T> tree)

  private BSTNode<T> removeNode(BSTNode<T> tree)

  private BSTNode<T> recRemove(T element, BSTNode<T> tree)

  public boolean remove (T element)

  private void inOrder(BSTNode<T> tree)

  private void preOrder(BSTNode<T> tree)

  private void postOrder(BSTNode<T> tree)

  public int reset(int orderType)

  public T getNext (int orderType)

  public void showStructure ( )

private void showSub ( BSTNode p, int level )

}
```