```
// Code for popping up a window that displays a custom component
// in this case we are displaying a Binary Search tree
// reference problem 4.38 of Weiss to compute tree node x,y positions
// input is a text file name that will form the Binary Search Tree
//     java DisplaySimpleTree textfile

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.io.*;
import java.util.*;

public class DisplaySimpleTree extends JFrame {
 JScrollPane scrollpane;
 DisplayPanel panel;

 public DisplaySimpleTree(MyTree t) {
  panel = new DisplayPanel(t);
  panel.setPreferredSize(new Dimension(300, 300));
  scrollpane = new JScrollPane(panel);
  getContentPane().add(scrollpane, BorderLayout.CENTER);
  setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  pack();  // cleans up the window panel
 }

 public static void main(String[] args) {
  MyTree t = new MyTree(); // t is Binary tree we are displaying
  BufferedReader diskInput;
  String word;
  // read in the words to create the Binary Search Tree
  if(args.length!=1){
          System.out.println("usage: java DisplayTree textfile");
    System.exit(0);
  }
  try { //reads in words from a file
   diskInput = new BufferedReader(new InputStreamReader(
     new FileInputStream(
             new File(args[0])));// file name is on command line
   Scanner input=new Scanner(diskInput);
   while (input.hasNext()) {
    word=input.next();
    word=word.toLowerCase(); // use lower case only
    t.root = t.insert(t.root, word);  //insert word into Binary Search Tree
    t.inputString= t.inputString + " " + word; // add word to input string
   }
  }
  catch (IOException e) {
    System.out.println("io exception");
   }

  t.computeNodePositions(); //finds x,y positions of the tree nodes
  t.maxheight=t.treeHeight(t.root); //finds tree height for scaling y axis
  DisplaySimpleTree dt = new DisplaySimpleTree(t);//get a display panel
  dt.setVisible(true); //show the display
 }
}
```

```
class DisplayPanel extends JPanel {
  MyTree t;
  int xs;
  int ys;

  public DisplayPanel(MyTree t) {
    this.t = t; // allows dispay routines to access the tree
    setBackground(Color.white);
    setForeground(Color.black);
  }

  protected void paintComponent(Graphics g) {
    g.setColor(getBackground()); //colors the window
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(getForeground()); //set color and fonts
    Font MyFont = new Font("SansSerif",Font.PLAIN,10);
    g.setFont(MyFont);
    xs=10;   //where to start printing on the panel
    ys=20;
    g.drawString("Binary Search tree for the input string:\n",xs,ys);
    ys=ys+10;;
    int start=0;
    //  print input string on panel, 150 chars per line
    // if string longer than 23 lines don't print
    if(t.inputString.length()<23*150){
        while((t.inputString.length()-start)>150){
          g.drawString(t.inputString.substring(start,start+150),xs,ys);
          start+=151;
          ys+=15;
        }
        g.drawString(t.inputString.substring(start,t.inputString.length()),xs,ys);
    }
    MyFont = new Font("SansSerif",Font.BOLD,20); //bigger font for tree
    g.setFont(MyFont);
    this.drawTree(g, t.root); // draw the tree
    revalidate(); //update the component panel
  }
```

```
    public void drawTree(Graphics g, Node root) {//actually draws the tree
      int dx, dy, dx2, dy2;
      int SCREEN_WIDTH=800; //screen size for panel
      int SCREEN_HEIGHT=700;
      int XSCALE, YSCALE;
      XSCALE=SCREEN_WIDTH/t.totalnodes; //scale x by total nodes in tree
     YSCALE=(SCREEN_HEIGHT-ys)/(t.maxheight+1); //scale y by tree height

    if (root != null) { // inorder traversal to draw each node
      drawTree(g, root.left); // do left side of inorder traversal
      dx = root.xpos * XSCALE; // get x,y coords., and scale them
      dy = root.ypos * YSCALE +ys;
      String s = (String) root.data; //get the word at this node
      g.drawString(s, dx, dy); // draws the word
// this draws the lines from a node to its children, if any
      if(root.left!=null){ //draws the line to left child if it exists
        dx2 = root.left.xpos * XSCALE;
        dy2 = root.left.ypos * YSCALE +ys;
        g.drawLine(dx,dy,dx2,dy2);
      }
      if(root.right!=null){ //draws the line to right child if it exists
        dx2 = root.right.xpos * XSCALE;//get right child x,y scaled position
        dy2 = root.right.ypos * YSCALE + ys;
        g.drawLine(dx,dy,dx2,dy2);
      }
      drawTree(g, root.right); //now do right side of inorder traversal
    }
   }
}

  class MyTree {
    String inputString= new String();
    Node root;
    int totalnodes = 0; //keeps track of the inorder number for horiz. scaling
    int maxheight=0;//keeps track of the depth of the tree for vert. scaling

    MyTree() {
     root = null;
    }

    public int treeHeight(Node t){
            if(t==null) return -1;
        else return 1 + max(treeHeight(t.left),treeHeight(t.right));
    }
    public int max(int a, int b){
              if(a>b) return a; else return b;
    }

    public void computeNodePositions() {
     int depth = 1;
     inorder_traversal(root, depth);
    }
```

```
//traverses tree and computes x,y position of each node, stores it in the node

   public void inorder_traversal(Node t, int depth) {
    if (t != null) {
     inorder_traversal(t.left, depth + 1); //add 1 to depth (y coordinate)
     t.xpos = totalnodes++; //x coord is node number in inorder traversal
     t.ypos = depth; // mark y coord as depth
     inorder_traversal(t.right, depth + 1);
    }
}

/* below is standard Binary Search tree insert code, creates the tree */

   public Node insert(Node root, String s) { // Binary Search tree insert
    if (root == null) {
     root = new Node(s, null, null);
     return root;
    }
    else {
     if (s.compareTo((String)(root.data)) == 0) {
       return root;  /* duplicate word  found - do nothing */
     } else   if (s.compareTo((String)(root.data)) < 0)
                root.left = insert(root.left, s);
              else
                root.right = insert(root.right, s);
     return root;
    }
   }
 }

class Node {  //standard Binary Tree node
   Object data;
   Node left;
   Node right;
   int xpos;  //stores x and y position of the node in the tree
   int ypos;

   Node(String x, Node l, Node r) {
    left = l;
    right = r;
    data = (Object) x;
   }
 }
```