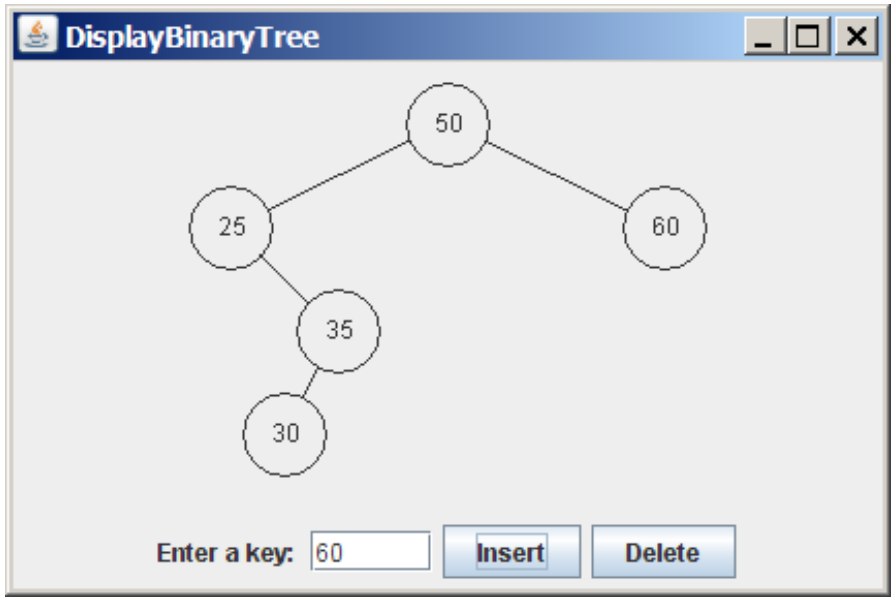
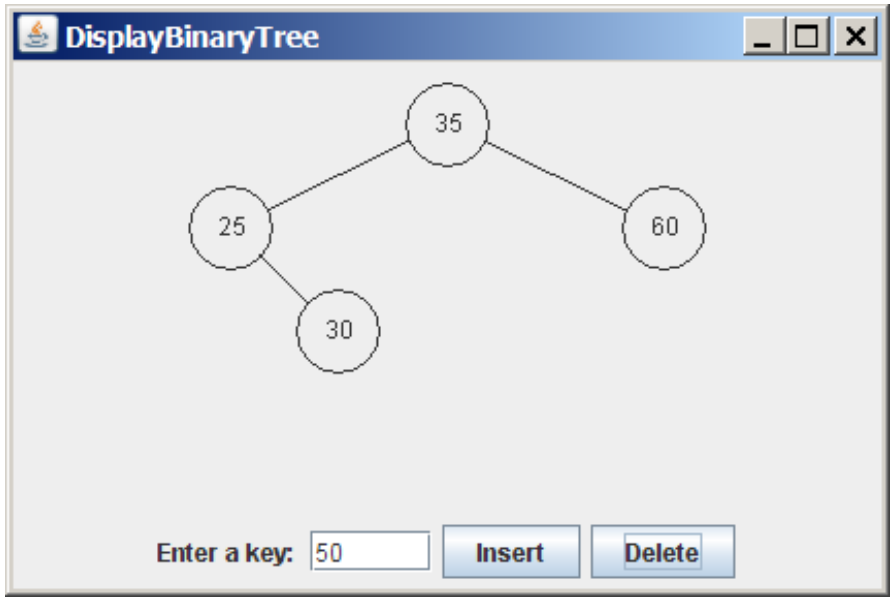
Tree Visualization

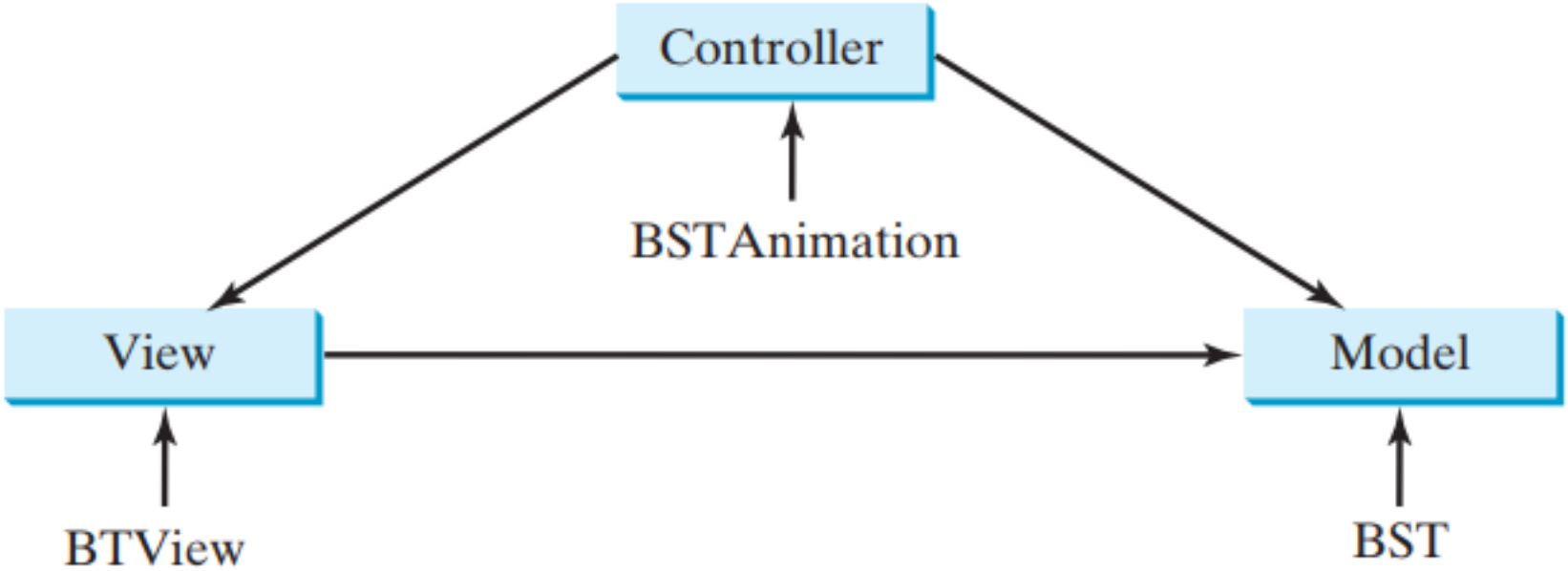
- It is a recursive structure, so you can display a binary tree using recursion
  - Display the root, then display the two subtrees recursively





Tree Visualization

- Tree visualization is an example of the ***model-view-controller*** (MVC) software architecture
  - The model is for storing and handling data
  - The view is for visually presenting the data
  - The controller handles the user interaction with the model and controls the view



```java
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;

public class BSTAnimation extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    BST<Integer> tree = new BST<>(); // Create a tree

    BorderPane pane = new BorderPane();
    BTView view = new BTView(tree); // Create a View
    pane.setCenter(view);

    TextField tfKey = new TextField();
    tfKey.setPrefColumnCount(3);
    tfKey.setAlignment(Pos.BASELINE_RIGHT);
    Button btInsert = new Button("Insert");
    Button btDelete = new Button("Delete");
    HBox hBox = new HBox(5);
    hBox.getChildren().addAll(new Label("Enter a key: "),
            tfKey, btInsert, btDelete);
    hBox.setAlignment(Pos.CENTER);
    pane.setBottom(hBox);
```

```
    btInsert.setOnAction(e -> {
      int key = Integer.parseInt(tfKey.getText());
      if (tree.search(key)) { // key is in the tree already
        view.displayTree();
        view.setStatus(key + " is already in the tree");
      } else {
        tree.insert(key); // Insert a new key
        view.displayTree();
        view.setStatus(key + " is inserted in the tree");
      }
    });
    btDelete.setOnAction(e -> {
      int key = Integer.parseInt(tfKey.getText());
      if (!tree.search(key)) { // key is not in the tree
        view.displayTree();
        view.setStatus(key + " is not in the tree");
      } else {
        tree.delete(key); // Delete a key
        view.displayTree();
        view.setStatus(key + " is deleted from the tree");
      }
    });
    // Create a scene and place the pane in the stage
    Scene scene = new Scene(pane, 450, 250);
    primaryStage.setTitle("BSTAnimation"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
  public static void main(String[] args) {
    launch(args);
  }}
```

```java
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;
import javafx.scene.text.Text;

public class BTView extends Pane {
  private BST<Integer> tree = new BST<>();
  private double radius = 15; // Tree node radius
  private double vGap = 50; // Gap between two levels in a tree

  BTView(BST<Integer> tree) {
    this.tree = tree;
    setStatus("Tree is empty");
  }

  public void setStatus(String msg) {
    getChildren().add(new Text(20, 20, msg));
  }

  public void displayTree() {
    this.getChildren().clear(); // Clear the pane
    if (tree.getRoot() != null) {
      // Display tree recursively
      displayTree(tree.getRoot(), getWidth() / 2, vGap, getWidth() / 4);
    }
  }
```

```
/** Display a subtree rooted at position (x, y) */
private void displayTree(BST.TreeNode<Integer> current,
    double x, double y, double hGap) {
  if (current.left != null) {
    // Draw a line to the left node
    getChildren().add(new Line(x - hGap, y + vGap, x, y));
    // Draw the left subtree recursively
    displayTree(current.left, x - hGap, y + vGap, hGap / 2);
  }
  if (current.right != null) {
    // Draw a line to the right node
    getChildren().add(new Line(x + hGap, y + vGap, x, y));
    // Draw the right subtree recursively
    displayTree(current.right, x + hGap, y + vGap, hGap / 2);
  }
  // Display the current node
  Circle circle = new Circle(x, y, radius);
  circle.setFill(Color.WHITE);
  circle.setStroke(Color.BLACK);
  getChildren().addAll(circle,
     new Text(x - 4, y + 4, current.element + ""));
}
}
```