```
/*
 * Countries of the World App 1.0
 * CountryDataTable.java "CountryData Table"
 * Waleed Gudah
 *
 */

import java.text.DecimalFormat;
import java.util.Arrays;

public class CountryDataTable {

    private int rootPtr = -1;
    private int n = 0; // number of good records (insert ++) (delete --)
    private int nextEmpty = 0; // next empty location (insert ++)
    private TheLog tLog;
    private bstNode[] countryDataTable;
    private int nodesVisited; // Number of visited nodes

    public CountryDataTable(TheLog tLog) {

        countryDataTable = new bstNode[0];

        rootPtr = -1;

        n = 0;

        nextEmpty = 0;

        this.tLog = tLog;
    }

    //
******************************************************************************************//
    // This method guarantees the node was inserted before writing to
    // the log//
    public void insert(String key, String name, String continent, int area,
            int population, float lifeExpectancy) {

        addNode(key, name, continent, area, population, lifeExpectancy);

        if (!key.isEmpty()) {

            DecimalFormat formatter = new DecimalFormat("#,###.##");

            if (name.length() > 18) {

                name = name.substring(0, 18);

            }

            tLog.transProcess(String.format("    " + key
                    + " %-18s %-13s %10s %13s  %3s ", name, continent,
                    formatter.format(area), formatter.format(population),
                    lifeExpectancy));

        }
        tLog.transProcess("   OK, country inserted");

        tLog.transProcess("        >> " + nodesVisited + " nodes visited");

    }

    //
******************************************************************************************//
    public void addNode(String key, String name, String continent, int area,
            int population, float lifeExpectancy) {

        bstNode newNode = new bstNode(key, name, continent, area, population,
                lifeExpectancy);

        countryDataTable = Arrays.copyOf(countryDataTable,
                countryDataTable.length + 1);

        countryDataTable[nextEmpty] = newNode;

        nodesVisited = 0;

        int i;

        int parentI = 0;

        char LorR = '\0';

        if (rootPtr == -1) { // [special case - no nodes in BST yet]

            rootPtr = nextEmpty;

        }

        else // [normal case]
        {
            i = rootPtr;

            while (i != -1) {

                parentI = i;

                if (name.compareToIgnoreCase(countryDataTable[i].getName()
                        .trim()) < 0) {

                    nodesVisited++;

                    i = countryDataTable[i].getLeft();

                    LorR = 'L';

                } else {

                    i = countryDataTable[i].getRight();

                    LorR = 'R';
```

```java
                }

            }

            if (LorR == 'L') {

                countryDataTable[parentI].setLeft(nextEmpty);

            } else {

                countryDataTable[parentI].setRight(nextEmpty);

            }

        }

        n++;

        nextEmpty++;

    }

    // ***************************************************************//
    // echo, guarantees node exists before sending to log************//
    public boolean deleteNode(String name) {

        boolean success = false;

        bstNode node = binarySearch(name);

        if (node != null) {

            node.makeTomb();

            n--;

            tLog.transProcess("      OK, country deleted");

            success = true;

        }

        return success;
    }

    // ***************************************************************//
    // echo, guarantees node exists before sending to log************//
    public void selectNode(bstNode current) {

        if (current != null) {

            String adjustName = current.getName();

            DecimalFormat formatter = new DecimalFormat("#,###.##");

            if (current.getName().length() > 18) {
```

```
            adjustName = current.getName().substring(0, 18);

        }
        tLog.transProcess(String.format("  " + current.getKey()

        + " %-18s %-13s %10s %13s " + current.getLifeExpectancy(),

        adjustName, current.getContinent(),
                formatter.format(current.getArea()),
                formatter.format(current.getPopulation())));

    }

    tLog.transProcess("        >> " + nodesVisited + " nodes visited");

}

// ***************************************************************//
public bstNode binarySearch(String target) {
    int i;

    nodesVisited = 0;

    bstNode searchNode = null;

    i = rootPtr;

    if (countryDataTable[0].getName().trim()
            .equalsIgnoreCase(target.trim())) {

        return countryDataTable[0];

    }

    while ((!target.equalsIgnoreCase(countryDataTable[i].getName()))) {

        searchNode = countryDataTable[i];

        nodesVisited++;

        if (target.compareToIgnoreCase(countryDataTable[i].getName()) < 0) {

            i = countryDataTable[i].getLeft();

        } else {

            i = countryDataTable[i].getRight();

        }

        if (i == -1) {

            tLog.transProcess("   SORRY, invalid country name");

            return null;

        }
```

```
            // handle SUCCESSFUL search situation

            searchNode = countryDataTable[i];
        }
        return searchNode;

    }

    // ***************************************************************************************//
    // All nodes are visited in ascending order Recursion is used to go to one
    // node and ***//
    // then go to its child nodes and so forth ***//
    public void inOrderTraverse(int root) {

        DecimalFormat formatter = new DecimalFormat("#,###.##");

        if (root != -1) {

            inOrderTraverse(countryDataTable[root].getLeft());

            String adjust;

            adjust = countryDataTable[root].getName();

            if (adjust.length() > 18) {

                adjust = countryDataTable[root].getName().substring(0, 18);
            }
            tLog.toLog(String.format("      %-3s %-18s %-13s %10s %13s "

            + countryDataTable[root].getLifeExpectancy() + "\n",

            countryDataTable[root].getKey(), adjust,
                    countryDataTable[root].getContinent(),

                    formatter.format(countryDataTable[root].getArea()),

                    formatter.format(countryDataTable[root].getPopulation())));

            inOrderTraverse(countryDataTable[root].getRight());
        }

    }

    //
******************************************************************************************//
    public int getNodesVisited() {

        return nodesVisited;

    }

    // ***************************************************************//
    public bstNode[] getList() {

        return countryDataTable;
    }

    // ***************************************************************//
    public int getNumberOfNodes() {

        return n;

    }

    // ***************************************************************//
    public void finishUp(boolean sS) {

        if (sS == true) {

            tLog.snapShot(countryDataTable, n, nextEmpty, rootPtr);

        }

        tLog.statusCode("CountryDataTable Finished");
    }
    // ***************************************************************//
}
```