```java
package Drugs;
import com.sun.xml.internal.bind.v2.model.core.ID;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.util.Scanner;
import java.io.IOException;

/** this is the main function Drug bank where all
 * the operations are being performed upon the drug
 *
 */

public class drugBank {
    FileWriter writeFile;
    Drug[] data;
    BinaryNode root;

    public drugBank() {
        ReadData();
    }

    /** this class reads the data and stores it into
     * a file and keeps the storage as an array of drugs
     *
     */

    public void ReadData() {
        try {
            File file = new File("dockedApproved.tab");
            Scanner scanner = new Scanner(file);
            String text = scanner.nextLine();
            int counter = 0;
            while (scanner.hasNextLine()) {
                text = scanner.nextLine();
                counter++;
            }
            scanner.close();
            data = new Drug[counter];
            file = new File("dockedApproved.tab");
            scanner = new Scanner(file);
            String[] store_array;
            text = scanner.nextLine();

            for (int i = 0; i < data.length; i++) {
                store_array = scanner.nextLine().split("\\t");
                data[i]=new Drug(store_array[0].trim(),store_array[1].trim(),store_array[2].trim(),store_array[3].trim(),store_array[4].trim(),Double.parseDouble(store_array[5].trim()));
            }
        } catch (FileNotFoundException e) {
            System.out.println("No file found!");
            e.printStackTrace();
        }
    }

    /** the create class just creates the drug item
     * into the tree for the bank
     *
     */

    public void create() {
        for (int i = 0; i < data.length; i++) {
            root = insert(root, data[i]);
        }
    }

    /** the insert class tries to insert the nodes
     * that are being entered or read from the file
     * to their desired positions
     *
     * @param rt
     * @param drug
     * @return
     */

    public BinaryNode insert(BinaryNode rt, Drug drug
) {
        if (rt == null) {
            rt = new BinaryNode(null, drug,null);
        } else {
            if (drug.drugBankId.compareToIgnoreCase(rt.item.drugBankId) < 0) {
                rt.left = insert(rt.left, drug);
            }else {
                rt.right = insert(rt.right, drug);
            }
        }
        return rt;
    }

    /** the inorder traverse basically
     * makes use of this traversal to print all
     * nodes in ascending order of drugs
     *
     */

    public void inOrderTraverse() {
        try {
            writeFile = new FileWriter("dockedApprovedSorted.tab");
            traverseinorder(root, writeFile);
            writeFile.flush();
            writeFile.close();
        } catch (IOException e) {
            System.out.println("error occurred");
            e.printStackTrace();
        }

    }

    public void traverseinorder(BinaryNode b, FileWriter tem) {
        if (b == null) {
            return;
        } else {
            traverseinorder(b.left, tem);
            try {
                tem.write(b.item.drugBankId + "" + b.item.GenericName + "" + b.item.drugGroups + "" + b.item.score + "\n");
            } catch (IOException e) {
                System.out.println("error occurred again");
                e.printStackTrace();
            }
            traverseinorder(b.right, tem);
        }
    }

    /** the search function just searches the desired
     * drug from the tree.
     *
     * @param Db_id
     * @return
     */

    public BinaryNode search(String Db_id) {
        BinaryNode tem = root;
        while(tem.item.drugBankId.compareToIgnoreCase(Db_id) !=0){
            if (tem.item.drugBankId.compareToIgnoreCase(Db_id) > 0) {
                tem = tem.left;
            } else {
                tem = tem.right;
            }
            if (tem == null) {
                System.out.println("No such drug found");
                return null;
            }
        }


        return tem;
        }


    /** the delete fucntion deletes the drug
     * from the file
     *
     * @param rt
     * @param id
     * @return
     */

        public BinaryNode delete(BinaryNode rt, String id) {
        if (rt ==  null) {
            return rt;
        }
        if(id.compareToIgnoreCase(rt.item.drugBankId) <0) {
            rt.left = delete(rt.left, id);
        }
        else if(id.compareToIgnoreCase(rt.item.drugBankId) >0) {
            rt.right = delete(rt.right, id);
        }

       else if (rt.left != null && rt.right != null) {
           rt.item = find_min(rt.right).item;
           rt.right = delete(rt.right, rt.item.drugBankId);



    }
       else {
           rt = (rt.left != null) ? rt.left : rt.right;
        }
        return rt;

    }

    public BinaryNode find_min(BinaryNode rt) {
            if (rt!=null){
                while(rt.left!=null) {
                    rt =rt.left;
                }
            }
            return  rt;
    }

    /** the depth function calculates the depth of the
     * tree for the drugs
     *
     * @param DbId
     * @return
     */

    public int Depth(String DbId) {
        int counter = 0;
        BinaryNode tem = root;
        while (tem.item.drugBankId.compareToIgnoreCase(DbId) > 0){
            if (tem.item.drugBankId.compareTo(DbId) != 0) {
                tem = tem.left;
                counter = counter + 1;
            } else {
                tem = tem.right;
                counter = counter + 1;

            }
        }


        return counter;
    }

    public int Depth1(BinaryNode r) {
        int x;
        int y;
        if (r == null) {
            return 0;
        }
        else{
            x = Depth1(r.left);
            y = Depth1(r.right);

            if (x > y) {
                return x+1;
            }
            else{
                return y+1;
            }
        }
    }

    /** the main function is to basically tests all
     * the functions with the drug ids to see whether
     * the functions are properly working or not.
     *
     * @param args
     */

    public static void main(String[] args) {
        drugBank db = new drugBank();

        db.create();
        db.inOrderTraverse();
        try{
            FileWriter f = new FileWriter("Output.txt");
            int calcDepth = db.Depth("DB01050");
            System.out.println(("the depth is: " + calcDepth));
            f.write("the depth of the node is: " + calcDepth+"\n");

            int deepestNode = db.Depth1(db.root);

            System.out.println((" the depth is :" + deepestNode));
            f.write("the depth is: " + deepestNode+
"\n");
            BinaryNode search = db.search("DB00316");
            if (search != null) {
                System.out.println(("the drug has been found"));
                f.write("drug has been found");
            }
            else{
                System.out.println(("the drug was not found"));
                f.write("the drug was not found in the file" + "\n");
            }

            BinaryNode delete = db.delete(db.root, "DB01065");
            if (delete != null) {
                System.out.println(("the drug has been deleted"));
                f.write("the drug has been deleted");

            }
            else{
                System.out.println(("the drug to be deleted does not exist"));
                f.write("the drug was not found");

            }
            f.flush();
            f.close();

            }
        catch (IOException e) {
            System.out.println("error");
        }
    }
}


```