

```
1 package Drugs;
2 import com.sun.xml.internal.bind.v2.model.core.ID;
3
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.io.FileWriter;
7 import java.util.Scanner;
8 import java.io.IOException;
9
10 /** this is the main function Drug bank where all
11 * the operations are being performed upon the drug
12 *
13 */
14
15 public class drugBank {
16     FileWriter writeFile;
17     Drug[] data;
18     BinaryNode root;
19
20     public drugBank() {
21         ReadData();
22     }
23
24     /** this class reads the data and stores it into
25     * a file and keeps the storage as an array of
26     drugs
27     *
28     */
29     public void ReadData() {
30         try {
31             File file = new File("dockedApproved.tab"
32 );
33             Scanner scanner = new Scanner(file);
34             String text = scanner.nextLine();
35             int counter = 0;
36             while (scanner.hasNextLine()) {
37                 text = scanner.nextLine();
38                 counter++;
39             }
40             scanner.close();
41         } catch (FileNotFoundException e) {
42             e.printStackTrace();
43         }
44     }
45 }
```

```

40         data = new Drug[counter];
41         file = new File("dockedApproved.tab");
42         scanner = new Scanner(file);
43         String[] store_array;
44         text = scanner.nextLine();
45
46         for (int i = 0; i < data.length; i++) {
47             store_array = scanner.nextLine().
split("\\t");
48             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()));
49         }
50     } catch (FileNotFoundException e) {
51         System.out.println("No file found!");
52         e.printStackTrace();
53     }
54 }
55
56 /** the create class just creates the drug item
57 * into the tree for the bank
58 *
59 */
60
61     public void create() {
62         for (int i = 0; i < data.length; i++) {
63             root = insert(root, data[i]);
64         }
65     }
66
67 /** the insert class tries to insert the nodes
68 * that are being entered or read from the file
69 * to their desired positions
70 *
71 * @param rt
72 * @param drug
73 * @return
74 */
75
76     public BinaryNode insert(BinaryNode rt, Drug drug

```

```

76 ) {
77     if (rt == null) {
78         rt = new BinaryNode(null, drug,null);
79     } else {
80         if (drug.drugBankId.compareToIgnoreCase(
rt.item.drugBankId) < 0) {
81             rt.left = insert(rt.left, drug);
82         }else {
83             rt.right = insert(rt.right, drug);
84         }
85     }
86     return rt;
87 }
88
89 /** the inorder traverse basically
90 * makes use of this traversal to print all
91 * nodes in ascending order of drugs
92 *
93 */
94
95 public void inOrderTraverse() {
96     try {
97         writeFile = new FileWriter("
dockedApprovedSorted.tab");
98         traverseinorder(root, writeFile);
99         writeFile.flush();
100        writeFile.close();
101    } catch (IOException e) {
102        System.out.println("error occurred");
103        e.printStackTrace();
104    }
105
106 }
107
108 public void traverseinorder(BinaryNode b,
FileWriter tem) {
109     if (b == null) {
110         return;
111     } else {
112         traverseinorder(b.left, tem);
113         try {

```

```

114         tem.write(b.item.drugBankId + "" + b
        .item.GenericName + "" + b.item.drugGroups + "" + b.
        item.score + "\n");
115     } catch (IOException e) {
116         System.out.println("error occurred
        again");
117         e.printStackTrace();
118     }
119     traverseinorder(b.right, tem);
120 }
121 }
122
123     /** the search function just searches the
        desired
124         * drug from the tree.
125         *
126         * @param Db_id
127         * @return
128         */
129
130     public BinaryNode search(String Db_id) {
131         BinaryNode tem = root;
132         while(tem.item.drugBankId.
        compareToIgnoreCase(Db_id) !=0){
133             if (tem.item.drugBankId.
        compareToIgnoreCase(Db_id) > 0) {
134                 tem = tem.left;
135             } else {
136                 tem = tem.right;
137             }
138             if (tem == null) {
139                 System.out.println("No such drug
        found");
140                 return null;
141             }
142         }
143
144
145         return tem;
146     }
147

```

```

148
149     /** the delete fucntion deletes the drug
150     * from the file
151     *
152     * @param rt
153     * @param id
154     * @return
155     */
156
157     public BinaryNode delete(BinaryNode rt,
String id) {
158         if (rt == null) {
159             return rt;
160         }
161         if(id.compareToIgnoreCase(rt.item.drugBankId
) <0) {
162             rt.left = delete(rt.left, id);
163         }
164         else if(id.compareToIgnoreCase(rt.item.
drugBankId) >0) {
165             rt.right = delete(rt.right, id);
166         }
167
168         else if (rt.left != null && rt.right != null
) {
169             rt.item = find_min(rt.right).item;
170             rt.right = delete(rt.right, rt.item.
drugBankId);
171
172
173
174     }
175     else {
176         rt = (rt.left != null) ? rt.left : rt.
right;
177     }
178     return rt;
179
180 }
181
182     public BinaryNode find_min(BinaryNode rt) {

```

```

183         if (rt!=null){
184             while(rt.left!=null) {
185                 rt =rt.left;
186             }
187         }
188         return  rt;
189     }
190
191     /** the depth function calculates the depth of
192     the
193     * tree for the drugs
194     * 
195     * @param DbId
196     * @return
197     */
198     public int Depth(String DbId) {
199         int counter = 0;
200         BinaryNode tem = root;
201         while (tem.item.drugBankId.
compareToIgnoreCase(DbId) > 0){
202             if (tem.item.drugBankId.compareTo(DbId
) != 0) {
203                 tem = tem.left;
204                 counter = counter + 1;
205             } else {
206                 tem = tem.right;
207                 counter = counter + 1;
208             }
209         }
210     }
211
212
213     return counter;
214 }
215
216 public int Depth1(BinaryNode r) {
217     int x;
218     int y;
219     if (r == null) {
220         return 0;

```

```

221     }
222     else{
223         x = Depth1(r.left);
224         y = Depth1(r.right);
225
226         if (x > y) {
227             return x+1;
228         }
229         else{
230             return y+1;
231         }
232     }
233 }
234
235 /** the main function is to basically tests all
236 * the functions with the drug ids to see
whether
237 * the functions are properly working or not.
238 *
239 * @param args
240 */
241
242 public static void main(String[] args) {
243     drugBank db = new drugBank();
244
245     db.create();
246     db.inOrderTraverse();
247     try{
248         FileWriter f = new FileWriter("Output.
249         int calcDepth = db.Depth("DB01050");
250         System.out.println(("the depth is: " +
251         calcDepth));
252         f.write("the depth of the node is: " +
253         calcDepth+"\n");
254
255         int deepestNode = db.Depth1(db.root);
256         System.out.println((" the depth is : " +
257         deepestNode));
258         f.write("the depth is: " + deepestNode+

```

```
256 "\n");
257         BinaryNode search = db.search("DB00316"
    );
258         if (search != null) {
259             System.out.println("the drug has
    been found");
260             f.write("drug has been found");
261         }
262         else{
263             System.out.println("the drug was
    not found");
264             f.write("the drug was not found in
    the file" + "\n");
265         }
266
267         BinaryNode delete = db.delete(db.root, "
    DB01065");
268         if (delete != null) {
269             System.out.println("the drug has
    been deleted");
270             f.write("the drug has been deleted"
    );
271         }
272         }
273         else{
274             System.out.println("the drug to be
    deleted does not exist");
275             f.write("the drug was not found");
276
277         }
278         f.flush();
279         f.close();
280
281     }
282     catch (IOException e) {
283         System.out.println("error");
284     }
285 }
286 }
287
288
```