# DictBinTree.java

1. /\* \* \* \* \* \* \* \* \* \* \* \* \* \*
2. \* Forfattere:
3. \* Josephine Søgaard Andersen, josea18@student.sdu.dk
4. \* Josias Kure, joulr18@student.sdu.dk
5. \* Kasper Jonassen, kajon18@student.sdu.dk
6. \* \* \* \* \* \* \* \* \* \* \* \* \* \*/
8. /\*
9. \* Her bliver de forskellige variabler deklareret.
10. \*/
11. public class DictBinTree implements Dict{
12. Node root;
13. int size;
14. int[] array;
15. int i;
17. /\*
18. \* Contructoren hvor der bliver lavet et ny dictionary, der er tomt og
19. \* hvor størrelsen er lig med 0.
20. \*/
21. public DictBinTree(){
22. this.root = null;
23. this.size = 0;
24. }
26. /\*
27. \* Metoden forudsætter at x ikke er lig med null. Herefter tager metoden først det venstre step,
28. \* hvor den sætter x.key til at være lig array[i], så vi gemmer vores x.key i arrayet.
29. \* Herefter tælles i én op og vælger højre siden.
30. \*/
31. public void treeWalk(Node x){
32. if(x != null){
33. treeWalk(x.left);
34. array[i] = x.key;
35. i++;
36. treeWalk(x.right);
37. }
38. }
40. /\*
41. \* Metoden løber igennem træet, startende ved roden, og returnerer elementets nøgle (k)
42. \*/
43. public Node treeSearch(Node x, int k){
44. if (x == null || k == x.key)
45. return x;
46. if (k < x.key)
47. return treeSearch(x.left, k);
48. else
49. return treeSearch(x.right, k);
50. }
52. /\*
53. \* Metoden bruger metoden treeSearch til at tjekke om elementet k findes i vores træ. Herefter returnes enten sandt eller falsk
54. \*/
55. public boolean search(int k){
56. if(treeSearch(root, k) != null){
57. return true;
58. }
59. return false;
60. }
62. /\*
63. \* Metoden returnerer en kopi af træets elementer i et array i sorteret orden
64. \*/
65. public int[] orderedTraversal(){
66. array = new int[size];
67. i = 0;
68. inOrderTreeWalk(root);
69. return array;
70. }
72. /\*
73. \* Metoden treeInsert indsætter træet T med nøglen z i træets dictionary
74. \*/
75. public void treeInsert(Node T, Node z){
76. Node y = null;
77. Node x = root;
79. while (x != null){
80. y = x;
81. if (z.key < x.key)
82. x = x.left;
83. else
84. x = x.right;
85. }
87. /\*
88. \* Hvis noden er tom, sætter vi z til at være roden. Hvis noden ikke er tom
89. \* placeres z afhængigt af værdien.
90. \*/
91. if (y == null)
92. root = z ;
93. else if (z.key < y.key)
94. y.left = z;
95. else
96. y.right = z;
97. }
99. /\*
100. \* Her bruges metoden treeInsert til at tælle size op hver gang der er
101. \* indsat et element i træet
102. \*/
103. public void insert(int k){
104. treeInsert(root, new Node(k));
105. size++;
106. }

109. /\*
110. \* Metoden krydser hver knude i træet rekrusivt. hvorefter den tæller i en op.
111. \*/
112. public int[] inOrderTreeWalk(Node x){
113. if (x != null){
114. inOrderTreeWalk(x.left);
115. array[i] = x.key;
116. i++;
117. inOrderTreeWalk(x.right);
118. }
119. return array;
120. }
122. class Node {
123. /\*
124. \* Vi har lavet en separat klasse til at repræsentere de forskellige knuder.
125. \*/
126. public int key;
127. public Node left;
128. public Node right;
130. /\*
131. \* Her indsættes nøglen k i hver Node samt sætter højre og venstre del af træet til null.
132. \*/
133. Node(int k){
135. this.key = k;
136. left = null;
137. right = null;
138. }
139. }
140. }

# Treesort.java

1. /\* \* \* \* \* \* \* \* \* \* \* \* \* \*
2. \* Forfattere:
3. \* Josephine Søgaard Andersen, josea18@student.sdu.dk
4. \* Josias Kure, joulr18@student.sdu.dk
5. \* Kasper Jonassen, kajon18@student.sdu.dk
6. \* \* \* \* \* \* \* \* \* \* \* \* \* \*/
8. import java.util.Scanner;
9. import java.util.ArrayList;
11. public class Treesort{
13. /\*
14. \* Vi laver en ArrayList hvor vi sætter vores tal ind fra scanneren, hvor while-lækken indsætter en værdi ind på hver plads i arrayet.
15. \*/
16. public static void main(String[] args) {
18. ArrayList<Integer> listOfNumbers = new ArrayList<Integer>();
19. Scanner scanner = new Scanner(System.in);
21. while (scanner.hasNextInt())
22. listOfNumbers.add(scanner.nextInt());
24. Dict dictonary = new DictBinTree();
26. /\*
27. \* Dette for-each loop kopierer alle elementer fra ListOfNumbers over i variablen number,
28. \* hvorefter den indsætter i dictionarien.
29. \*/
30. for (int number : listOfNumbers)
31. dictonary.insert(number);
33. /\*
34. \* Dette for-each loop kopierer alle elementer fra dictonary.orderedTraversal over i variablen number
35. \* Derefter udskrives værdien i terminalen
36. \*/
37. for (int number : dictonary.orderedTraversal())
38. System.out.println(number);
39. }
40. }