

# Morse Code als Baum

## Lösung

```
Application.java
1 package ch.bbw.pr.morsetree;
2
3 import ch.bbw.pr.morsetree.model.MorseTree;
4
5 /**
6  * Morse-Tree Application
7  * @author Peter Rutschmann
8  * @version 09.01.2020
9  */
10 public class Application {
11
12     public static void main(String[] args) {
13         System.out.println("Morse-Tree");
14         MorseTree myTree = new MorseTree();
15
16         System.out.println("Add some letters");
17         myTree.addLetter('e', ".");
18         myTree.addLetter('a', "-.");
19         myTree.addLetter('i', "..");
20         myTree.addLetter('t', "-");
21         myTree.addLetter('n', "-.");
22         myTree.addLetter('m', "--");
23
24         System.out.println();
25         System.out.println("Print the tree:");
26         myTree.traverse(myTree.getRoot());
27         System.out.println();
28         System.out.println("Decode one character");
29         System.out.println("decode -- " + myTree.decodeLetter("--"));
30         System.out.println();
31         System.out.println("Decode multiple characters");
32         String tmp = "--/./-/./.";
33         System.out.println("decode " + tmp + " ==> " + myTree.decode(tmp));
34     }
35 }
```

```

Application.java  Node.java
1 package ch.bbw.pr.morsetree.model;
2 /**
3  * Node
4  * @author Peter Rutschmann
5  * @version 09.01.2020
6  */
7 public class Node {
8     Node dot;
9     Node dash;
10    char letter;
11    String code;
12 }
13

```

```

Application.java  Node.java  MorseTree.java
1 package ch.bbw.pr.morsetree.model;
2
3 /**
4  * Morse-Tree
5  * @author Peter Rutschmann
6  * @version 09.01.2020
7  */
8 public class MorseTree {
9     private Node root;
10
11    public Node getRoot() {
12        return root;
13    }
14
15    public void addLetter(char letter, String code) {
16        if(root==null) {
17            root = new Node();
18            root.code="root";
19        }
20        addRecursive(root, letter, code, code);
21    }
22
23    private void addRecursive(Node current, char letter, String code, String codepart) {
24        if(codepart.length()==0) {
25            //codepart ist fertig ausgelesen, ich bin der Knoten
26            current.letter = letter;
27            current.code = code;
28        }
29        else if(codepart.charAt(0) == '.') {
30            //weiter mit dem dot-Knoten
31            if(current.dot == null) current.dot = new Node();
32            addRecursive(current.dot, letter, code, codepart.substring(1));
33        } else if(codepart.charAt(0) == '-') {
34            //weiter mit dem dash-Knoten
35            if(current.dash == null) current.dash = new Node();
36            addRecursive(current.dash, letter, code, codepart.substring(1));
37        } else {
38            //unbekanntes oder Trennzeichen
39            addRecursive(current, letter, code, codepart.substring(1));
40        }
41    }
42

```

```

42
43 public String decode(String code) {
44     String retVal="";
45     for(String part: code.split("/")) {
46         retVal = retVal + decodeLetter(root, part) + " ";
47     }
48     return retVal;
49 }
50
51 public Character decodeLetter(String code) {
52     if(root==null) {
53         return ' ';
54     }
55     return decodeLetter(root, code);
56 }
57
58 private Character decodeLetter(Node current, String codepart) {
59     if(codepart.length()==0) {
60         //codepart ist ausgelesen -> ich bin der Knoten
61         return current.letter;
62     }
63     else if(codepart.charAt(0) == '.') {
64         //weiter mit dem dot-Knoten
65         if(current.dot == null) return null;
66         return decodeLetter(current.dot, codepart.substring(1));
67     } else if(codepart.charAt(0) == '-') {
68         //weiter mit dem dash-Knoten
69         if(current.dash == null) return null;
70         return decodeLetter(current.dash, codepart.substring(1));
71     }
72     //Trennzeichen
73     return decodeLetter(current, codepart.substring(1));
74 }
75

```

```

75
76 public void traverse(Node node) {
77     if (node != null) {
78         System.out.println(node.letter + " " + node.code);
79         traverse(node.dot);
80         traverse(node.dash);
81     }
82 }
83
84 }

```