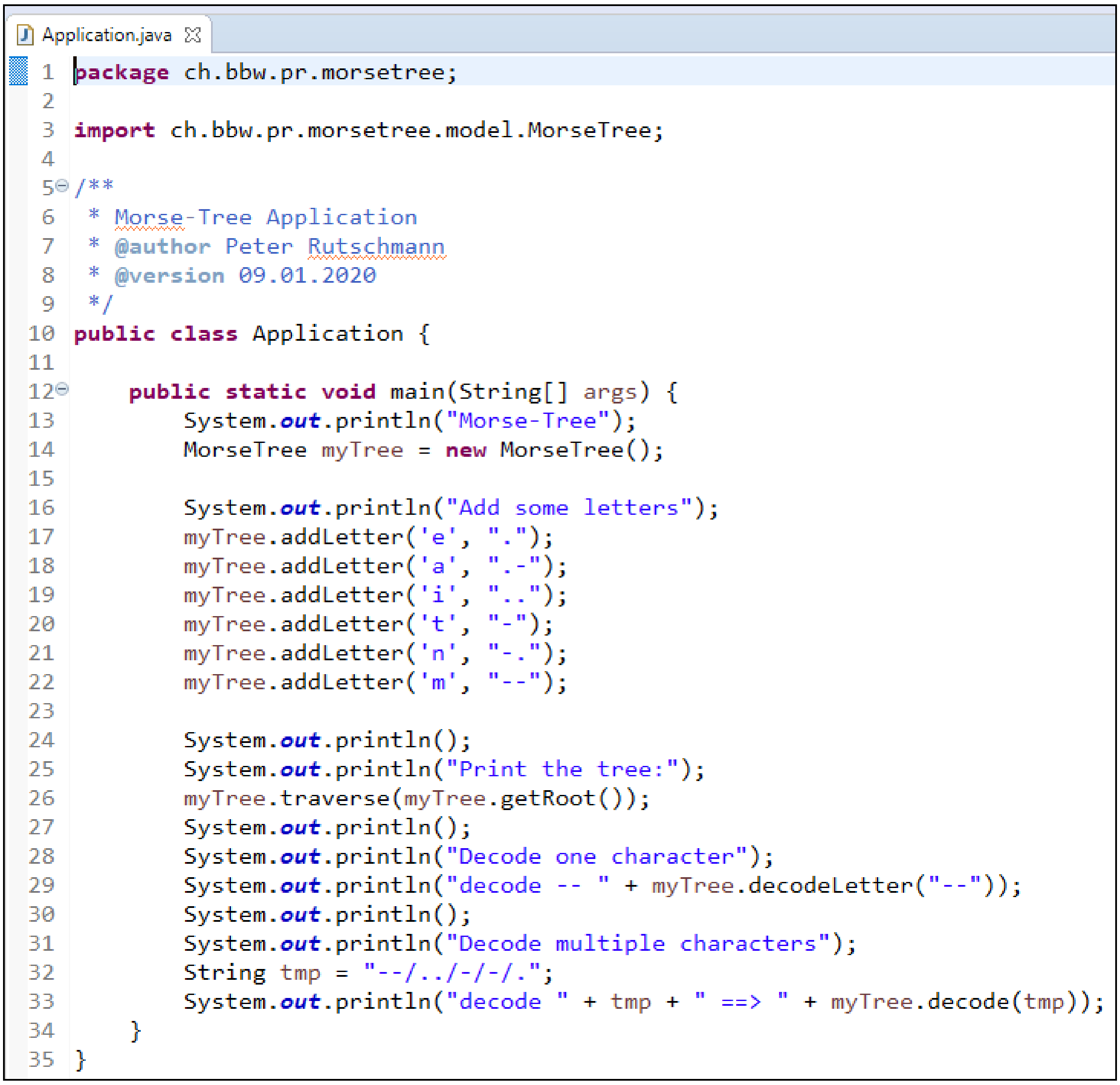
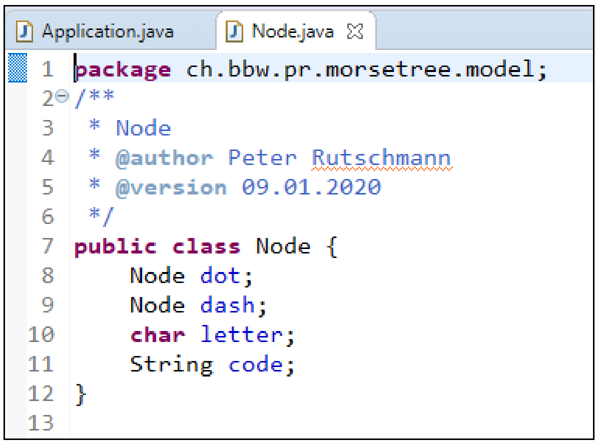
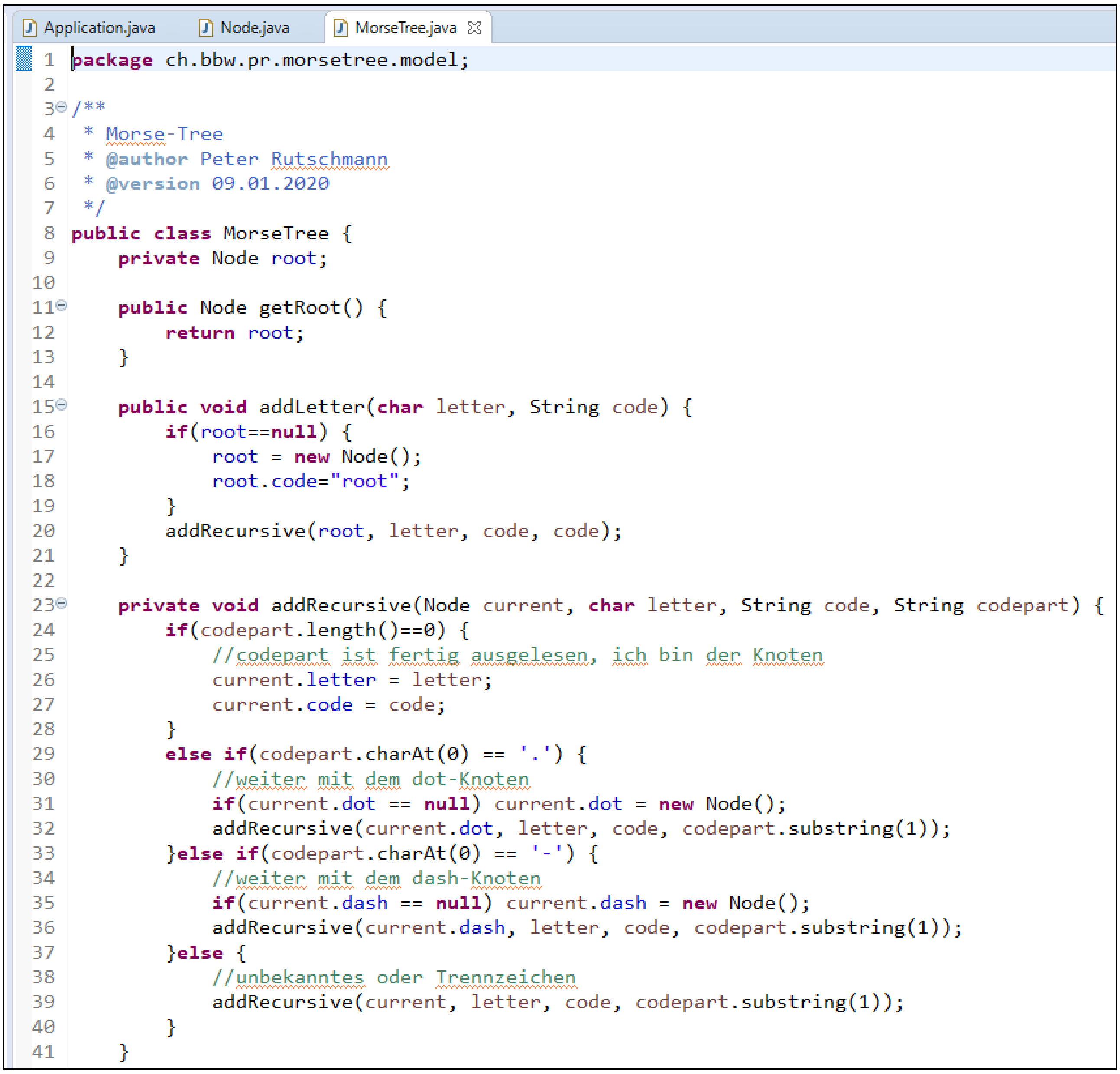
Morse Code als Baum
Lösung


Application.java

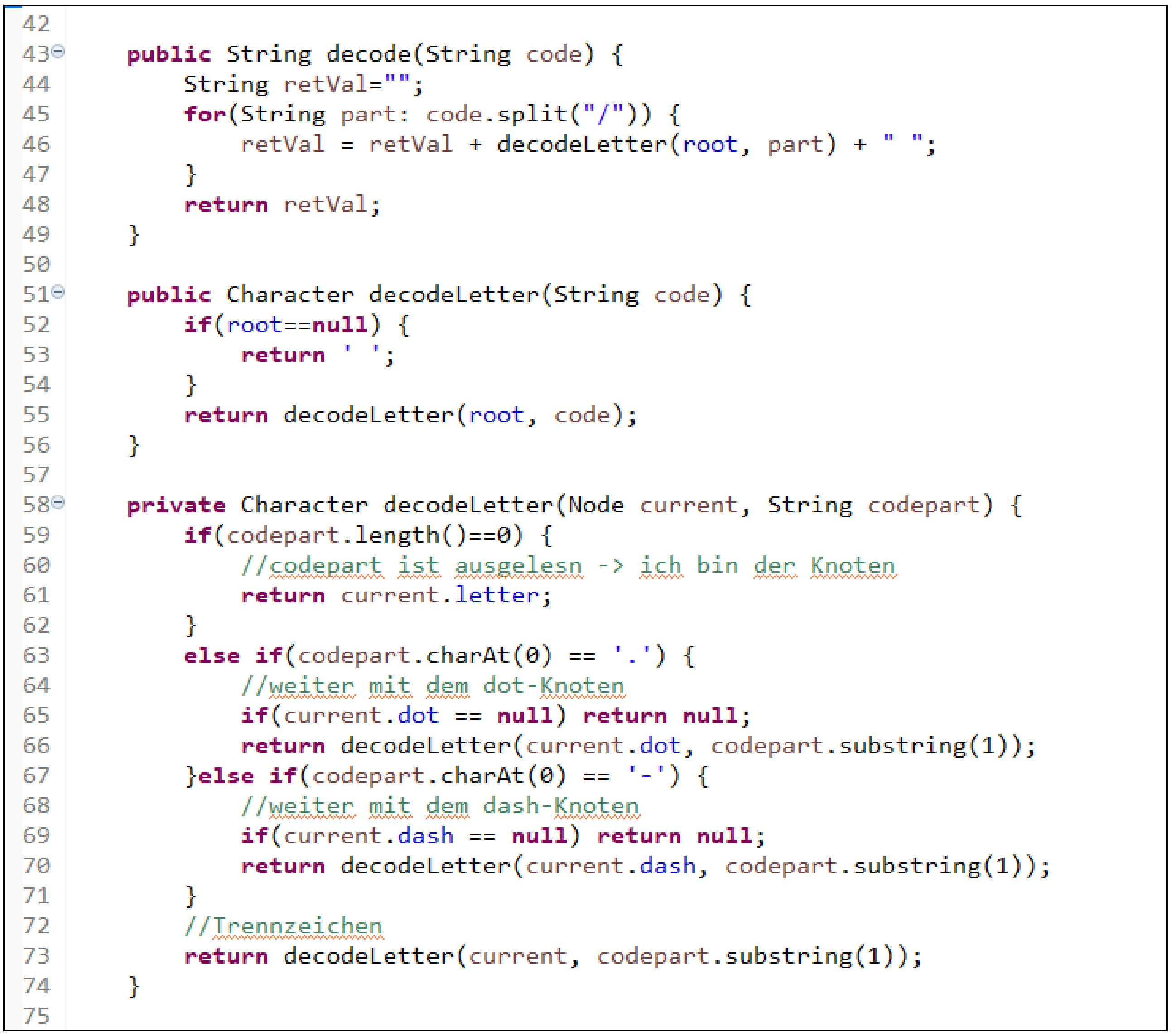
```java
package ch.bbw.pr.morsetree;

import ch.bbw.pr.morsetree.model.MorseTree;

/**
 * Morse-Tree Application
 * @author Peter Rutschmann
 * @version 09.01.2020
 */
public class Application {

    public static void main(String[] args) {
        System.out.println("Morse-Tree");
        MorseTree myTree = new MorseTree();

        System.out.println("Add some letters");
        myTree.addLetter('e', ".");
        myTree.addLetter('a', ".-");
        myTree.addLetter('i', "..");
        myTree.addLetter('t', "-");
        myTree.addLetter('n', "-.");
        myTree.addLetter('m', "--");

        System.out.println();
        System.out.println("Print the tree:");
        myTree.traverse(myTree.getRoot());
        System.out.println();
        System.out.println("Decode one character");
        System.out.println("decode -- " + myTree.decodeLetter("--"));
        System.out.println();
        System.out.println("Decode multiple characters");
        String tmp = "--/../-/-/.";
        System.out.println("decode " + tmp + " ==> " + myTree.decode(tmp));
    }
}
```



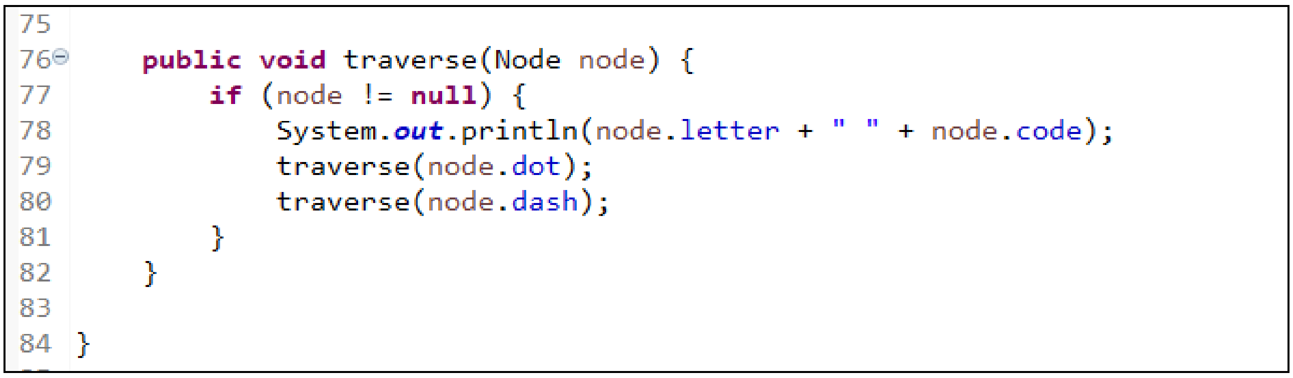
```java
package ch.bbw.pr.morsetree.model;
/**
 * Node
 * @author Peter Rutschmann
 * @version 09.01.2020
 */
public class Node {
    Node dot;
    Node dash;
    char letter;
    String code;
}

```



```java
package ch.bbw.pr.morsetree.model;

/**
 * Morse-Tree
 * @author Peter Rutschmann
 * @version 09.01.2020
 */
public class MorseTree {
    private Node root;

    public Node getRoot() {
        return root;
    }

    public void addLetter(char letter, String code) {
        if(root==null) {
            root = new Node();
            root.code="root";
        }
        addRecursive(root, letter, code, code);
    }

    private void addRecursive(Node current, char letter, String code, String codepart) {
        if(codepart.length()==0) {
            //codepart ist fertig ausgelesen, ich bin der Knoten
            current.letter = letter;
            current.code = code;
        }
        else if(codepart.charAt(0) == '.') {
            //weiter mit dem dot-Knoten
            if(current.dot == null) current.dot = new Node();
            addRecursive(current.dot, letter, code, codepart.substring(1));
        }else if(codepart.charAt(0) == '-') {
            //weiter mit dem dash-Knoten
            if(current.dash == null) current.dash = new Node();
            addRecursive(current.dash, letter, code, codepart.substring(1));
        }else {
            //unbekanntes oder Trennzeichen
            addRecursive(current, letter, code, codepart.substring(1));
        }
    }
```



```java

    public String decode(String code) {
        String retVal="";
        for(String part: code.split("/")) {
            retVal = retVal + decodeLetter(root, part) + " ";
        }
        return retVal;
    }

    public Character decodeLetter(String code) {
        if(root==null) {
            return ' ';
        }
        return decodeLetter(root, code);
    }

    private Character decodeLetter(Node current, String codepart) {
        if(codepart.length()==0) {
            //codepart ist ausgelesn -> ich bin der Knoten
            return current.letter;
        }
        else if(codepart.charAt(0) == '.') {
            //weiter mit dem dot-Knoten
            if(current.dot == null) return null;
            return decodeLetter(current.dot, codepart.substring(1));
        }else if(codepart.charAt(0) == '-') {
            //weiter mit dem dash-Knoten
            if(current.dash == null) return null;
            return decodeLetter(current.dash, codepart.substring(1));
        }
        //Trennzeichen
        return decodeLetter(current, codepart.substring(1));
    }

```



```java

    public void traverse(Node node) {
        if (node != null) {
            System.out.println(node.letter + " " + node.code);
            traverse(node.dot);
            traverse(node.dash);
        }
    }

}
```