Morsecodierer und Morsedecodierer

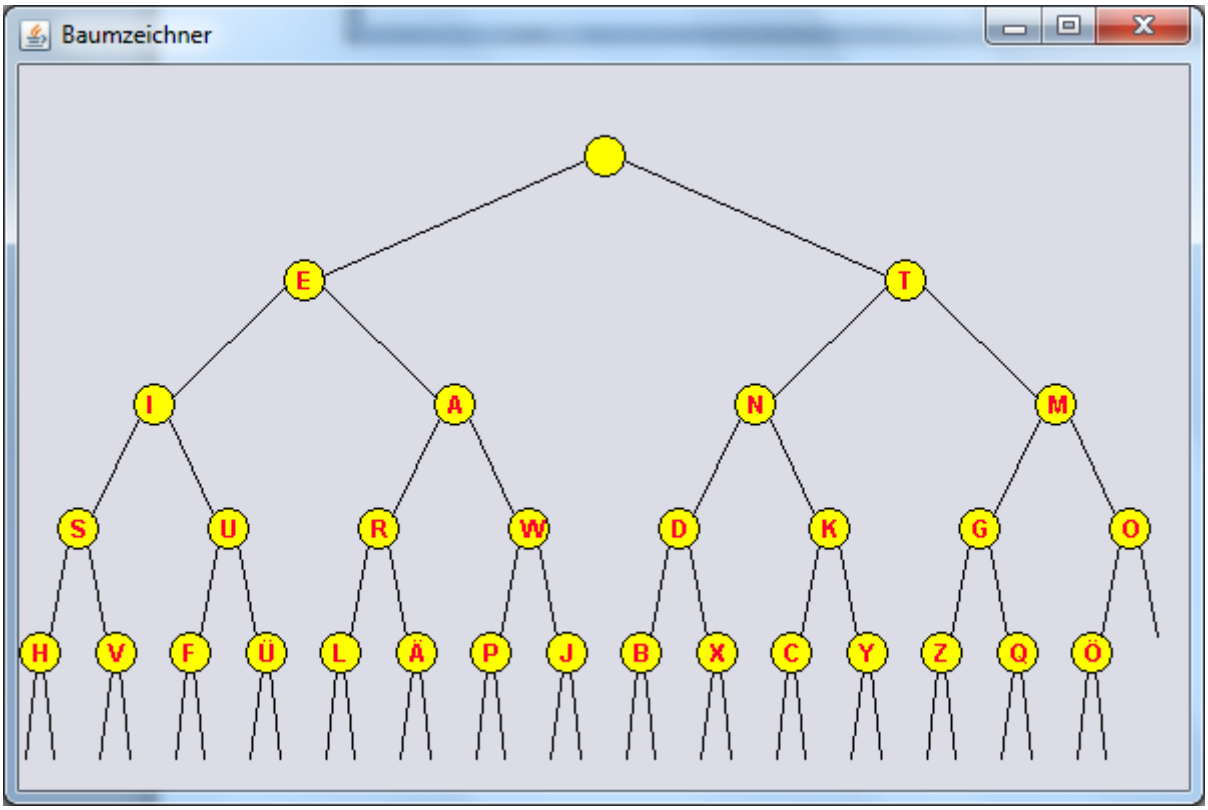
Ziel des Projektes
Entwicklung eines Programms, mit dem Texte codiert und decodiert werden können.

Morse hat bei seiner Codierung Buchstaben, die sehr häufig im Textvorkommen, durch kurze Zeichenfolgen codiert und Zeichen, die seltener vorkommen, durch längere Zeichenfolgen.

Darstellung des Morsecodes als Tabelle

A ·-	F ····	K -·-	P ·--·	U ··-	Z --··
B -···	G --·	L ·-··	Q --·-	V ···-	Ä ·-·-
C -·-·	H ····	M --	R ·-·	W ·--	Ö ---·
D -··	I ··	N -·	S ...	X -··-	Ü ··--
E ·	J ·---	O ---	T -	Y -·--	CH ----

Darstellung des Morsecodes als Binärbaum

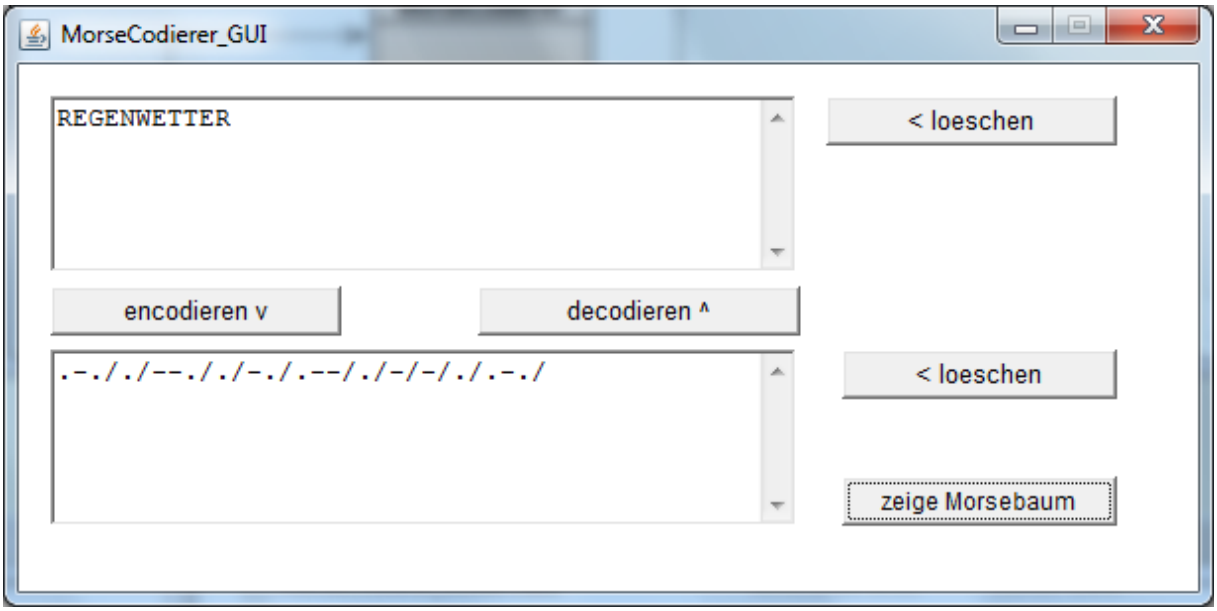


In dem Binärbaum entspricht eine Kante nach links einem Punkt und eine Kante nach rechts einem Strich.

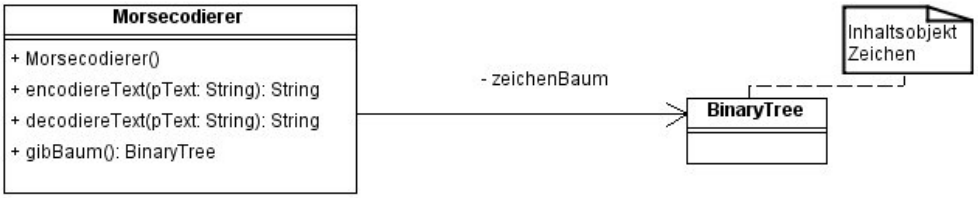
Vergleich der Darstellungen (Tabelle/ Baum)
Die Baumdarstellung bietet eine Vorteil bei der Decodierung eines Morsetextes. Falls man sich auf den beschriebenen Zeichensatz beschränkt, erhält man in wenigen Suchschritten (Die Anzahl der Suchschritten entspricht der Länge des codierten Zeichens.) den zugehörigen Buchstaben. Beim Codieren des Morsetextes ergibt sich kein Vorteil, da im schlimmsten Fall der Baum volständig durchsucht werden muss.

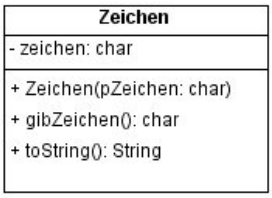
Die Tabellendarstellung könnte man in einem Array oder einer Liste realisieren. Hier sind beim Codieren und beim Decodieren bedeutend mehr Suchschritte erforderlich.

Benutzungsoberfäche des Morsecodierers (und –decodierers) (mit zusätzlichen, schon implementierten Möglichkeiten, die das Testen erleichtern)



Der Morsecodierer (und –decodierer) soll mit Hilfe der Klasse **`BinaryTree`** realisiert werden.





Die Klasse **`Morsecodierer`**

Dokumentaion der öffentlichen Methoden

Konstruktor **`Morsecodierer()`**

Ein Objekt der Klasse wird erzeugt, ein Morsebaum wird erzeugt. Dazu wird der Morsecode für jedes Zeichen aus einer Datei geladen.

Anfrage **`String encodiereText(String pText)`**

Der als Parameter übergebene Text **`pText`** wird buchstabenweise in den Morsecode übersetzt, die Punkt-Strich-Folgen, die zu einem Buchstaben gehören, werden durch **`'/'`** getrennt.

Anfrage **`String decodiereText(String pText)`**

Der als Parameter übergeben Morsetext besteht aus Punkten, Strichen und dem Trennzeichen **`'/'`**. Die zwischen den Trennzeichen liegenden Punkt-Strich-Folgen werden gemäß dem Morsecode in Buchstaben übersetzt.

Anfrage **`BinaryTree<Zeichen> gibBaum()`**

Der Morsebaum wird gegeben. Die Methode soll dem Baumzeichner ermöglichen, den Baum darzustellen.

```
43
44  private void fuegeEin(char pZeichen, String pCode) {
45    zeichenBaum = fuegeEin(pZeichen, pCode, zeichenBaum);
46  }
47
48  private BinaryTree<Zeichen> fuegeEin(char pZeichen, String pCode, BinaryTree<Zeichen> pTree){
49    Zeichen z = new Zeichen(pZeichen);
50    if (pTree.isEmpty()) {
51      pTree = new BinaryTree<Zeichen>(z);
52    }
53    if (pCode.equals("")) {
54      pTree.setContent(z);
55    } else {
56      if (pCode.charAt(0)== '.') {
57        pTree.setLeftTree(fuegeEin(pZeichen, pCode.substring(1), pTree.getLeftTree()));
58      } else {
59        if (pCode.charAt(0) == '-') {
60          pTree.setRightTree(fuegeEin(pZeichen, pCode.substring(1), pTree.getRightTree()));
61        }
62        else {
63          System.out.println("Fehler!");
64        }
65      }
66    }
67    return pTree;
68  }
69
```

Die Methoden **`fuegeEin`** fügt abhängig vom Morsecode ein Zeichen in den Morsebaum ein. Liegt ein Punkt vor, so wird ein linker Teilbaum angefügt, liegt ein Strich vor, wird ein rechter Teilbaum angefügt. Der Parameter pCode wird jeweils um das erste Zeichen verkürzt. Der Morsebaum ist erst korrekt, wenn die alle Zeichen eingetragen worden sind. Zwischemzeitlich können in verschiedenen Wurzeln gleiche Zeichen stehen.

```
113   private char decodiereZeichen(String pMorsecode) {
114     BinaryTree<Zeichen> bin = zeichenBaum;
115     String code = pMorsecode;
116     while (code.length() > 0) {
117       if (code.charAt(0) == '.')
118         bin = bin.getLeftTree();
119       else
120       if (code.charAt(0) == '-')
121         bin = bin.getRightTree();
122       else
123         System.out.println("Fehler2");
124       code = code.substring(1);
125     }
126     Zeichen z = bin.getContent();
127     return z.gibZeichen();
128   }
```

Ist das erste Zeichen des Codes ein Punkt, so wird der linke Teilbaum zugewiesen, falls das Zeichen ein Strich ist, wird der rechte Teilbaum zugewiesen. Bei dem zu bearbeitenden Code wird das ersten Zeichen entfernt, bis die Zeichenkette leer ist. Wenn alle Zeichen des Morsecodes abgearbeitet sind, enthält die Wurzel des Teilbaums den zum Morsecode zugehörigen Buchstaben. Dieser wird dann zurückgegeben.

```
82   private String encodiereZeichen(char pZeichen, BinaryTree<Zeichen> pBaum, String pTeilcode) {
83     if (pBaum.isEmpty()) {
84       return "";
85     }
86     String ergebnis = "";
87     if ((pBaum.getContent()).gibZeichen() == pZeichen) {
88       ergebnis =  pTeilcode;
89     }
90     else {
91       ergebnis = encodiereZeichen(pZeichen, pBaum.getLeftTree(), pTeilcode + ".") +
92                  encodiereZeichen(pZeichen, pBaum.getRightTree(), pTeilcode + "-");
93     }
94     return ergebnis;
95   }
```

In dieser Methode wird der Morsebaum nach dem Buchstaben, der codiert werden soll, rekursiv durchsucht. Für jede Verweigung nach links, wir ein Punkt an die Zeichenkette pTeilcode gehängt, für eine Verzweigung nach rechts wird ein Strich an den den Teilcode gehängt.In pTeilcode steht jeweils der Morsecode zu dem Zeichen, das in der aktuellen Wurzel abgespeichert ist. Wird der Buchstabe im Baum gefunden, so wird der Teilcode zurück geliefert. Für alle anderen Buchstaben wird eine leere Zeichenkette zurück gegeben. Der Morsebaum wird für jeden Buchstaben vollständig durchlaufen.

```
70  public String encodiereText(String pText) {
71    pText=pText.toUpperCase();
72    String code = "";
73    char zeichen;
74    for (int zaehler = 0; zaehler <= pText.length()-1; zaehler++) {
75      zeichen = pText.charAt(zaehler);
76      code = code + encodiereZeichen(zeichen,zeichenBaum,"");
77      code = code + '/';
78    }
79    return code;
80  }
```

Die Methode **`encodiereText`** bestimmt buchstabenweise den Morsecode und ergänzt jeweils ein Trennzeichen.

```
97  public String decodiereText(String pText) {
98    String rest = pText;
99    String morse;
100   String text = "";
101   int index;
102   do {
103     index = rest.indexOf('/');
104     morse = rest.substring(0,index);
105     rest = rest.substring(index+1,rest.length());
106     if (morse.length() > 0)
107       text = text + decodiereZeichen(morse);
108   } while (rest.length()>0);
109   return text;
110 }
```

Die Methode **`decodiereText`** bestimmt den Morsecode für einen Buchstaben, der zwischen zwei Trennzeichen liegt. Mit Hilfe der Methode **`decodiereZeichen`** wird dann der Buchstabe zum Morsecode bestimmt.