Aufgabe 4: Auto-Scrabble

Team: „Was ist JAVA?“

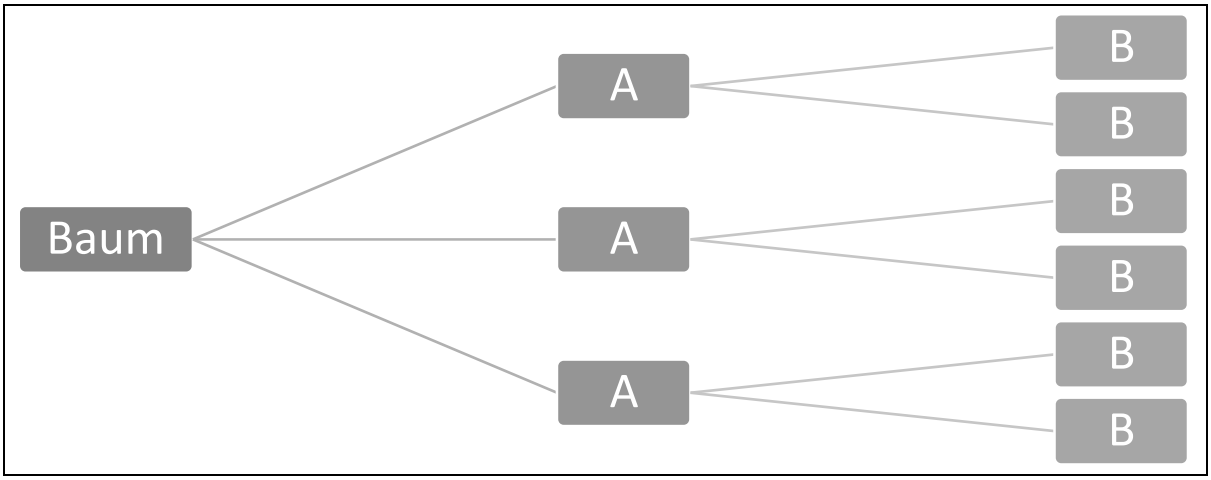
Team-ID: 00107

24. November 2017

Inhaltsverzeichnis

Lösungsidee

Unsere Idee für die Umsetzung der Auto-Scrabble Aufgabe war es einen Baum mit unterschiedlich vielen Verzweigungen, sogenannten Knoten zu erzeugen (siehe unten).



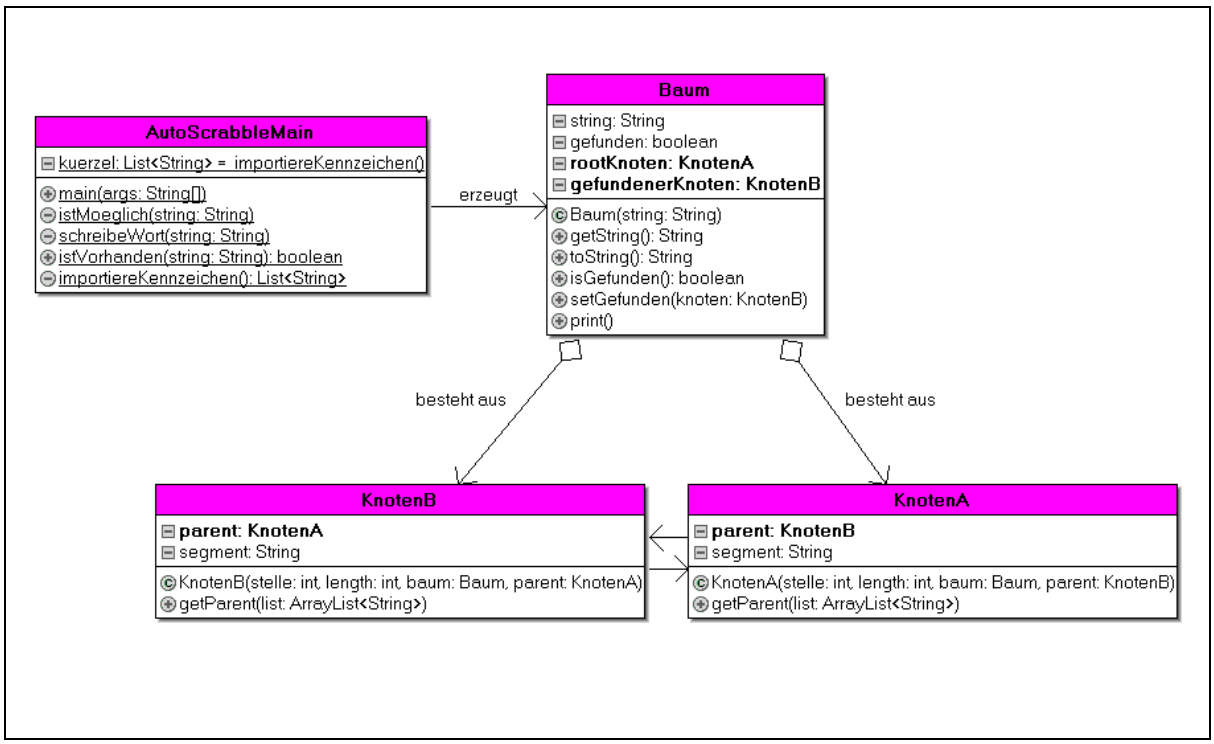
Dabei war zu beachten, dass es zwei unterschiedliche Formen von Knoten geben kann:

Zum einen gibt es den A-Knoten, der den Teil des Kennzeichens repräsentiert, der das Kürzel der Stadt oder des Landkreises enthält. Er kann aus einem, zwei oder drei Buchstaben bestehen. Zum anderen gibt es den B-Knoten, der den Teil des Kennzeichens repräsentiert, der den/die Buchstaben enthält, die der Besitzer frei wählen kann. Dieser Teil eines Kennzeichens besteht aus einem oder zwei Buchstaben.

Der Anfangsknoten ist ein A-Knoten, dieser ruft nacheinander die B-Knoten auf. Die B-Knoten rufen jeweils wieder A-Knoten auf und so weiter. Nach jedem Aufrufen eines A-Knotens muss jedoch überprüft werden, ob das Kürzel der Stadt oder des Landkreises auch wirklich existiert. Nach jedem Erstellen eines B-Knoten wird ermittelt, ob das Wortende erreicht wurde. Ist dies der Fall, kann das geprüfte Wort mit Kennzeichen dargestellt werden.

Umsetzung

Wir haben unsere Lösungsidee in einem objektorientierten Programm in der Programmiersprache Java umgesetzt:



Unser Programm besteht aus insgesamt vier Klassen, drei davon sind aktiv am Erstellen und Überprüfen der Kennzeichen beteiligt. Hingegen stellt die AutoScrabbleMain Klasse die Kürzelliste bereit und erzeugt einen neuen Baum für jedes Wort.

Die Baum-Klasse erstellt den ersten A-Knoten (rootKnoten), stellt Informationen an die Knoten bereit und speichert den untersten Knoten, mit dem später der Weg über die verschiedenen Knoten herausgefunden werden kann und somit auch das Wort geschrieben werden kann.

Die A-Knoten Klasse repräsentiert den vorderen Teil des Kennzeichens. Wenn sie aufgerufen wird, wird geprüft, ob der Kennzeichenabschnitt (1, 2 oder 3 Buchstaben) in der Kürzelliste (gespeichert in der Main-Klasse) vorhanden ist. Um Ressourcen zu sparen werden erst die unteren Knoten erzeugt, bevor weitere A-Knoten auf der gleichen Ebene erstellt werden.

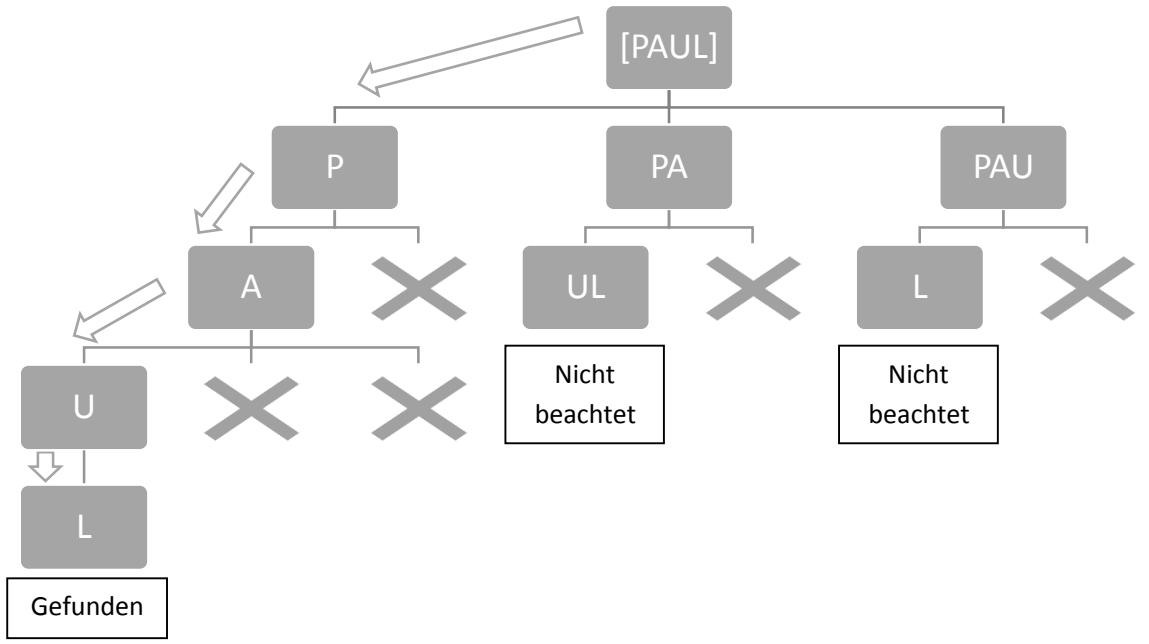
Die B-Konten Klasse repräsentiert den hinteren Teil eines Kennzeichens. Die B-Konten-Klasse prüft zuerst, ob das Ende des Wortes erreicht wurde und setzt gegebenenfalls eine Markierung in der Baum-Klasse, die verhindert, dass weitere Knoten erstellt werden.

Wenn das Erstellen der A und B-Konten nicht zu einem Setzen einer Abbruch-Markierung in der Baum Klasse geführt hat ist davon auszugehen, dass das Wort nicht darzustellen ist. Wenn der letzte B-Knoten eine Markierung in der Baum Klasse setzt, wird auch seine Speicheradresse abgespeichert. So kann die

Baum Klasse feststellen, ob das Wort darstellbar ist und den Weg über die Knoten rekonstruieren, um das Wort später darstellen zu können.

Beispiele

Das Programm kann Wörter aller Länge darstellen. Als Beispiel wird hier mein Name dargestellt:



Die Darstellung wurde vereinfacht und die Überprüfung der Kennzeichenkürzel vernachlässig. Die Überprüfung der Kennzeichenkürzel wird bei jedem Erstellen eines neuen A-Knoten ausgeführt.

Wie zu sehen, werden immer die kleinsten Möglichkeiten ausgewählt. Um Ressourcen zu sparen, werden die anderen Darstellungsmöglichkeiten nicht mehr berücksichtigt, wenn eine Kombination erst einmal gefunden wurde.

Zwar sieht es in der Darstellung so aus, doch der ersten Ebene befindet sich nur ein A-Knoten, der drei verschiedene Zustände annehmen kann. Das Gleiche gilt für die folgenden Knoten. So befinden sich in der zweiten Ebene z.B. drei Konten usw. Damit können eine unnötige Verwendung von Speicherplatz vermieden werden.

Quellcode

```
public class Baum {

        private String string;

        private boolean gefunden = false;


        private KnotenA rootKnoten;
        private KnotenB gefundenerKnoten;


        public Baum(String string) {
```

```
// Speichert den übergebenen String,
der Dargestellt werden soll.
// Markierung, ob schon eine
Möglichkeit der Darstellung gefunden
wurde.
// Speichert den ersten/root Knoten.
// Speichert den letzten B-Konten der
für das Schreiben wichtig ist.
```

```
                this.string = string;
                rootKnoten = new KnotenA(0,string.length()
                - 1, this, null);
        }


        public String getSubstring(int stelle1, int
        stelle2) {
                return this.string.substring(stelle1,
                stelle2);
        }

        public boolean isGefunden() {
                return this.gefunden;
        }

        public void setGefunden(KnotenB knoten) {
                this.gefunden = true;
                this.gefundenerKnoten = knoten;
        }

        public void print() {
                if (gefunden) {
                java.util.ArrayList<String> list = new
                java.util.ArrayList<String>();
                this.gefundenerKnoten.getParent(list);
                list.trimToSize();
                for (int i = 0; i < list.size(); i += 2) {
                System.out.print(list.get(list.size() - i
                - 1) + "-" + list.get(list.size() - i - 2)
                + " ");
                }
                } else {
                System.out.print("nicht Darstellbar");
                }
                System.out.println("");
        }

        @Override
        public String toString() {
                return "[" + string + "]";
        }

}
```

```
// Erstellt den ersten Knoten (stelle
= 0, länge = Stringlänge, die
Speicheradresse des Baums, null ->
kein übergeordneter Knoten)


// Funktion für die Knoten:
gibt einen Teil, des darzustellenden
Wortes, damit der Knoten ihn dann
überprüfen kann.


// Funktion für die Knoten; gibt
Auskunft, ob weitergesucht werden
soll.

// Setzen der Markierung; wird vom
letzten Konten aufgerufen, der sich
selbst übergibt und dann gespeichert
wird.

// schreibt die Kennzeichen, indem
eine Liste erstellt und von dem
untersten Knoten immer zum nächst
höheren gegeben wird und dabei
ergänzt wird
```

```
public class KnotenA {
        private KnotenB parent;
        private String segment;

        public KnotenA(int stelle, int length, Baum baum,
        KnotenB parent) {
                this.parent = parent;
                if (stelle + 1 <= length &&
                !baum.isGefunden()) {
                        if (AutoScrabbleMain.istVorhanden
                         (baum.getSubstring(
                        stelle, stelle + 1))) {
                        segment = baum.getSubstring(stelle,
                        stelle + 1);
                        new KnotenB(stelle + 1, length,
                        baum, this);
                        }
                }

                if (stelle + 2 <= length &&
                !baum.isGefunden()) {
                        if (AutoScrabbleMain.istVorhanden
                         (baum.getSubstring(
                        stelle, stelle + 2))) {
                        segment = baum.getSubstring(stelle,
                        stelle + 2);
                        new KnotenB(stelle + 2, length,
                        baum, this);
                        }
```

```
// repräsentiert den ersten Teil
eines Kennzeichens mit der Länge 1


// repräsentiert den ersten Teil
eines Kennzeichens mit der Länge 2
```

```
                }
                if (stelle + 3 <= length &&
                !baum.isGefunden()) {
                        if (AutoScrabbleMain.istVorhanden
                         (baum.getSubstring(
                        stelle, stelle + 3))) {
                        segment = baum.getSubstring(stelle,
                        stelle + 1);
                        new KnotenB(stelle + 3, length,
                        baum, this);
                        }
                }
        }

        public void getParent(java.util.ArrayList<String>
                list) {
                list.add(segment);
                if (parent != null) {
                        parent.getParent(list);
                }
        }
}
```

// repräsentiert den ersten Teil eines Kennzeichens mit der Länge 3

// wird zum Schreiben benötigt siehe Baum.print();

```
public class KnotenB {
        private KnotenA parent;
        private String segment;

        public KnotenB(int stelle, int length, Baum baum,
        KnotenA parent) {
                this.parent = parent;

                if (stelle + 0 == length &&
                !baum.isGefunden()) {
                        segment = baum.getSubstring(stelle,
                        stelle + 1);
                        baum.setGefunden(this);
                        return;
                }

                if (stelle + 1 == length &&
                !baum.isGefunden()) {
                        segment = baum.getSubstring(stelle,
                        stelle + 2);
                        baum.setGefunden(this);
                        return;
                }

                if (stelle + 1 < length &&
                !baum.isGefunden()) {
                        segment = baum.getSubstring(stelle,
                        stelle + 1);
                        new KnotenA(stelle + 1, length,
                        baum, this);
                }

                if (stelle + 2 < length &&
                !baum.isGefunden()) {
                        segment = baum.getSubstring(stelle,
                        stelle + 2);
                        new KnotenA(stelle + 2, length,
                        baum, this);
                }
        }

        public void getParent(java.util.ArrayList<String>
        list) {
                list.add(segment);
                if (parent != null) {
                        parent.getParent(list);
                }
        }
}
```

//prüft, ob die Länge des Wortes genau erreicht wurde; wobei der hintere Teil des Kennzeichens ein Buchstabe lang ist und setzt ggf. die Gefunden-Markierung

//prüft, ob die Länge des Wortes genau erreicht wurde; wobei der hintere Teil des Kennzeichens zwei Buchstaben lang ist und setzt ggf. die Gefunden-Markierung

//wenn die Länge mit einem hinteren Buchstaben noch nicht erreicht ist, wird ein weiterer Knoten erstellt

//wenn die Länge mit zwei hinteren Buchstaben noch nicht erreicht ist, wird ein weiterer Knoten erstellt

// wird zum Schreiben benötigt siehe Baum.print();

Lösungen der Aufgaben

Aufgabe 1

TI-MO kann nicht auf einem Kennzeichen stehen, da es kein Kürzel mit „TI“ gibt.
TIM-O kann nicht auf einem Kennzeichen stehen, da es kein Kürzel mit „TIM“ gibt.
T-I M-O kann nicht auf einem Kennzeichen stehen, da es kein Kürzel mit „T“ gibt.
Der Name Timo kann also wirklich nicht dargestellt werden.

Aufgabe 2

JAZZ, *UFO* und *OB* können nicht mit einem einzigen Kennzeichen dargestellt werden.

Aufgabe 3

Output:

```
[DONAUDAMPFSCHIFFFAHRTSKAPITAENSMUETZE] ist möglich
[BIBER] ist möglich
[CLINTON] ist möglich
[ETHERNET] ist möglich
[INFORMATIK] ist möglich
[LLANFAIRPWLLGWYNGYLLGOGERYCHWYRNDROBWLLLLANTYSILIOGOGOGOCH] ist möglich
[BUNDESWETTBEWERB] ist möglich
[RINDFLEISCHETIKETTIERUNGSUEBERWACHUNGSAUFGABENUEBERTRAGUNGSGESETZ] ist nicht möglich
[SOFTWARE] ist möglich
[TRUMP] ist möglich
[TSCHUESS] ist möglich
[VERKEHRSWEGEPLANUNGSBESCHLEUNIGUNGSGESETZ] ist möglich
fertig
```

Aufgabe 4

Output:

```
[DONAUDAMPFSCHIFFFAHRTSKAPITAENSMUETZE]
D-O N-AU D-A M-P F-S C-HI F-F F-A H-R TS-K AP-IT A-E N-S M-U E-T Z-E
[BIBER]
B-I B-ER
[CLINTON]
C-LI NT-ON
[ETHERNET]
E-T H-E R-N E-T
[INFORMATIK]
IN-FO R-M AT-IK
[LLANFAIRPWLLGWYNGYLLGOGERYCHWYRNDROBWLLLLANTYSILIOGOGOGOCH]
L-L A-N F-AI R-P W-L L-G W-Y N-GY L-L G-O G-E R-Y C-H W-Y R-N D-R OB-W L-L L-L AN-TY S-I L-I OG-O G-
O G-O C-H
[BUNDESWETTBEWERB]
B-U N-D E-S WE-TT B-E W-E R-B
[RINDFLEISCHETIKETTIERUNGSUEBERWACHUNGSAUFGABENUEBERTRAGUNGSGESETZ]
nicht Darstellbar
[SOFTWARE]
S-O F-T W-A R-E
[TRUMP]
TR-U M-P
[TSCHUESS]
TS-C H-U E-SS
[VERKEHRSWEGEPLANUNGSBESCHLEUNIGUNGSGESETZ]
V-E R-K E-H R-S W-E G-E P-L A-N UN-G S-B E-S C-H L-E UN-I G-U N-G S-G E-S E-TZ
fertig
```