Aufgabe 4: Auto-Scrabble

Team: Impromatik
Team-ID: 00061

27. November 2017

Inhaltsverzeichnis

Lösungsidee....1
Umsetzung....3
Beispiele....4
Quellcode....6

Lösungsidee

Die Idee der Lösung sollte hieraus vollkommen ersichtlich werden, ohne das auf die eigentliche Implementation Bezug genommen wird.

An der Lösung der ersten beiden Teilaufgaben werde ich meine Lösungsidee näher erläutern:

1. Stimmt es, dass TIMO nicht auf einem Kennzeichen stehen kann ?

   Ein Kennzeichen/Nummernschild besteht aus einem Kürzel, welches ein bis drei Buchstaben lang sein kann und einem Mittelteil, welcher ein bis zwei Buchstaben lang sein kann. Die darauffolgende Zahl wird ignoriert.

   So kann TIMO in [TI | MO] und [TIM | O] aufgteilt werden. Wenn die Kürzel der Nummernschilder existieren, kann das Wort auf einem Nummernschild stehen. Da “TI” und “TIM” nicht auf der Kürzelliste stehen, kann TIMO nicht auf einem Nummernschild stehen.

2. Gib ein weiteres Wort mit vier Buchstaben an, das nicht auf einem Kennzeichen stehen kann. Findest du sogar ein solches Wort mit drei Buchstaben? Mit zwei?

   Die ersten beiden und die ersten drei Buchstaben eines vier Buchstaben langen Wortes dürfen nicht auf der Kürzelliste enthalten sein, damit das Wort nicht auf einem Kennzeichen stehen kann. Man kann z.B. TIMO auch mit zwei Nummernschildern in [T | I] [M | O] aufteilen, die Aufgabenstellung spricht aber ausdrücklich von einem Nummernschild.

   Ich habe viele solcher Wörter gefunden wie z.B. TURM, IGEL und TANZ:

   “TU” und “TUR” existieren nicht als Kürzel weshalb TURM nicht auf Nummernschildern stehen kann.

   Es gibt auch drei Buchstaben lange Wörter, deren erster und erste beiden Buchstaben nicht auf der Kürzelliste stehen und welche deshalb nicht auf Kennzeichen stehen können wie z.B. ICH, TIM und OMA: “O” und “OM” stehen nicht auf der Kürzelliste, weshalb OMA nicht auf Nummernschildern stehen kann. Es existieren auch zwei Buchstaben lange Wörter wie z.B. IM, IN und UM, welche auch nicht auf Nummernschildern stehen können, da deren Anfangsbuchstabe nicht als Kürzel existiert.

3. Die dritte und vierte Teilaufgabe besagen, ein Programm zu schreiben, welches ein Wort prüft, ob es mit mehreren Kennzeichen geschrieben werden kann und jeweilige Kennzeichenfolgen ausgibt.

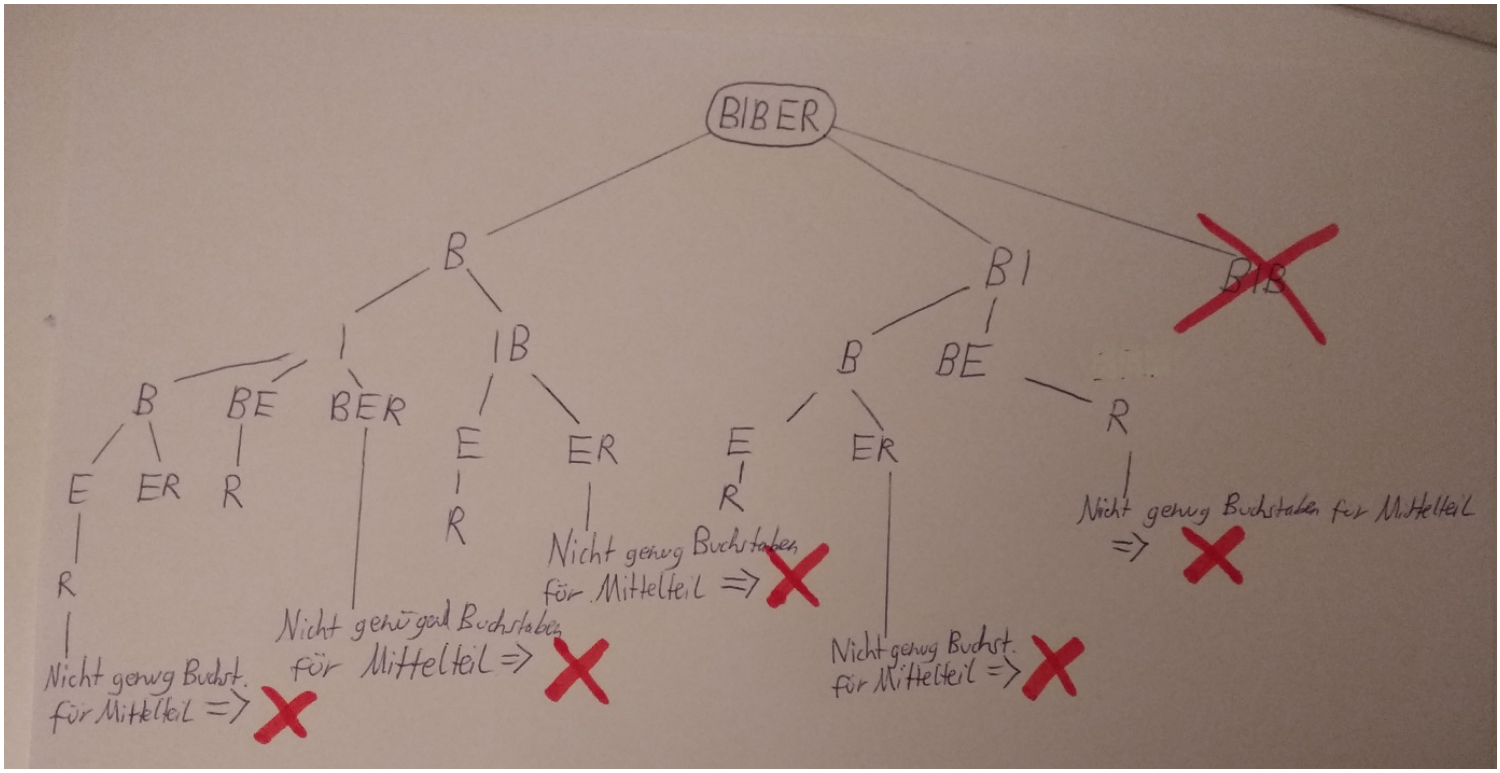
Aufgabe 4: Auto-Scrabble

Meine Lösungsidee war ein Algorithmus, der alle Aufteilungsmöglichkeiten für das zu prüfende Wort danach prüft, ob alle Kürzel auf der Kürzelliste stehen und zutreffende Aufteilungsmöglichkeiten ausgibt. Wenn Kennzeichenfolgen ausgegeben werden, kann das Wort auf Kennzeichen stehen und wenn nicht dann nicht. Für die Aufteilung des Wortes in mehrere Kennzeichen mit Kürzel und Mittelteil soll am Anfang des Wortes angefangen werden: Für das erste Kürzel gibt es drei Möglichkeiten(ein, zwei oder drei Buchstaben lang). Dann wird der Mittelteil zusammen mit dem Kürzel des nächsten Nummernschildes betrachtet, da er nicht darauf geprüft werden muss, ob er auf der Kürzelliste enthalten ist. Für die Fortsetzung jeder der anfangs drei Möglichkeiten gibt es somit sechs Möglichkeiten, da der Mittelteil 1-2 Zeichen lang sein kann und das Kürzel 1-3.

	Anzahl von Zeichen					
Kürzel	1		2		3	
Mittelteil	1	2	1	2	1	2

So ergeben sich wie in der Tabelle sichtbar 2 Möglichkeiten * 3 Möglichkeiten = 6 Möglichkeiten.

Aus jeder dieser 6 Möglichkeiten können sich wieder 6 Möglichkeiten für die weitere Aufteilung ergeben und aus diesen wieder 6 usw., bis das Programm am Wortende angekommen ist. Hier ein Baumdiagramm für die Aufteilungsmöglichkeiten des Wortes 'BIBER':



Jeder Pfad steht für eine Aufteilungsmöglichkeit. Sofern der Pfad mit einem X endet, kann die Aufteilungsmöglichkeit nicht auf einem Nummernschild stehen, da entweder das Kürzel nicht existiert wie bei 'BIB' oder nicht mehr genügend Buchstaben für den Mittelteil übrig sind.

Der Algorithmus beginnt am Wortanfang und prüft alle drei Möglichkeiten, ob die Kürzel auf der Kürzelliste stehen und noch genügend Buchstaben übrig sind. Für alle Möglichkeiten, bei denen dies der Fall ist, werden die nächsten sechs Möglichkeiten und für die Möglichen aus diesen sechs die nächsten sechs usw., bis das Wortende erreicht ist, geprüft. Sobald klar ist, dass die Aufteilungsmöglichkeit nicht auf einem Nummernschild stehen kann, wird aufgehört,

weiterzurechnen, und nur Pfade, welche ohne Fehler am Wortende angelangt sind wie [B | I] [B | ER] werden ausgegeben.

Umsetzung

Hier wird kurz erläutert, wie die Lösungsidee im Programm tatsächlich umgesetzt wurde. Hier können auch Implementierungsdetails erwähnt werden.

Die Lösungsidee wird in Java implementiert.

Als erstes wird die Kürzelliste aus der Datei kuerzelliste.txt mithilfe eines Scanners in eine ArrayList umgefüllt, weil wir dadurch die Methode [Name der ArrayList].contains(String arg0) nutzen können, welche wir verwenden, um zu prüfen, ob ein Kürzel in der Kürzelliste enthalten ist oder nicht.

Danach wird in if-Abfragen geprüft, ob der erste, die ersten beiden oder die ersten drei Buchstaben in der Kürzelliste enthalten sind und das Wort auch lang genug ist und wenn dies der Fall ist wird die Aufteilung des Wortes mit ab jetzt sechs Möglichkeiten pro Möglichkeit weiterberechnet:

Dazu wurde mithilfe der Implementierung der Thread-Klasse durch “implements Runnable” die ausführbare Klasse “Abzweigung” programmiert, die die nächsten Möglichkeiten prüft. Threads können gleichzeitig und ausserhalb der main-Methode ausgeführt werden, wodurch sich die Klasse selbst aufrufen und ausführen kann.

Beim Kreieren eines neuen Objekts der “Abzweigung”-Klasse muss wie im Konstruktor der Klasse festgelegt wurde das zu überprüfende Wort, die bisherige Position im Wort, ab der die nächsten Aufteilungsmöglichkeiten berechnet werden sollen und die bisherige Aufteilung des Worts, welche in einer ArrayList gespeicherd wird mitgegeben werden. Diese Informationen werden beim Ausführen benötigt.

Mit sechs if-Abfragen(einer pro Möglichkeit) wird geprüft, bei welchen der sechs Aufteilungsmöglichkeiten genug Buchstaben übrig sind und bei welchen die Kürzel in der Kürzelliste enthalten sind.

Dies geschieht mit `if (Autoscrabble.kuerzelliste.contains(kuezel)) { //Anweisung }`

Für alle Möglichkeiten, bei denen beides der Fall ist, wird die jeweilige Möglichkeit mit Mittelteil plus Kürzel der Liste mit der bisherigen Aufteilung hinzugefügt und ein neues Objekt der Abzweigung-Klasse kreiert und ausgeführt d.h. Die Klasse ruft und führt sich selber auf. Dabei wird das zu überprüfende Wort, die vorherige Position plus die Länge des hinzugefügten Mittelteils und Kürzels und die Liste, welcher der Mittelteil und das Kürzel hinzugefügt wurden weitergegeben und danach der Mittelteil und das Kürzel wieder aus der Liste entfernt, damit diese bei der nächsten der sechs if-Abfragen die Liste wie am Anfang ist.

Für jede Möglichkeit kann also ein neues Objekt erzeugt werden, das dann ab der Möglichkeit weiterrechnet, das alte Objekt wird danach gelöscht. Jedes Objekt steht also für eine Möglichkeit, welche gerade berechnet wird, und aus jedem Objekt können sechs neue entstehen.

Dies geschieht so lange, bis das Programm am Wortende angekommen ist d.h. Die Position des Objekts der Länge des zu bearbeiteten Wortes minus zwei oder drei gleicht(Bei der Position fangen wir bei 0 an zu zählen, der letzte Mittelteil kann 1 oder 2 Buchstaben lang sein).

Ob es am Ende angekommen ist, prüft das jeweilige Objekt direkt nach dem starten in zwei if-Abfragen, da es zwei Möglichkeiten für den letzten Mittelteil gibt. Wenn dies der Fall ist, wird der letzte Mittelteil zur Liste mit der bisherigen Aufteilung des Worts hinzugefügt und diese dann ausgegeben.

Es werden also alle Aufteilungsmöglichkeiten durchgegangen und nur die, die auch auf Kennzeichen stehen können werden ausgegeben.

Beispiele

Genügend Beispiele einbinden! Eigene Beispiele sind sehr gut! Und die Beispiele sollte diskutiert werden.

Wir rufen das Programm für Beispielwörter von der BwInf-Webseite auf:

```
Möglichkeiten für das Wort 'BIBER':
[B|I] [BE|R]
[B|I] [B|ER]
[BI|B] [E|R]
[B|IB] [E|R]
Das Wort kann auf einem Nummernschild stehen.

Möglichkeiten für das Wort 'BUNDESWETTBEWERB':
[B|UN] [D|ES] [WE|TT] [B|E] [WE|RB]
[B|UN] [D|ES] [WE|TT] [B|E] [WER|B]
[B|UN] [D|ES] [WE|TT] [B|EW] [ER|B]
[B|UN] [D|ES] [WE|TT] [B|EW] [E|RB]
[B|UN] [D|ES] [WE|TT] [B|E] [W|E] [R|B]
[B|UN] [D|E] [S|W] [E|TT] [B|E] [WER|B]
[B|UN] [D|E] [S|W] [E|TT] [B|EW] [E|RB]
[B|UN] [D|E] [S|W] [E|TT] [B|EW] [ER|B]
[B|UN] [D|E] [S|W] [E|TT] [B|E] [WE|RB]
[B|UN] [D|E] [S|W] [E|TT] [B|E] [W|E] [R|B]
[B|UN] [D|ES] [WE|TT] [BE|W] [E|RB]
[B|UN] [D|ES] [WE|TT] [BE|W] [ER|B]
[B|UN] [D|ES] [WE|TT] [BE|WE] [R|B]
[B|U] [N|D] [E|SW] [E|TT] [BE|W] [E|RB]
[B|U] [N|D] [ES|W] [E|TT] [B|EW] [E|RB]
[B|U] [N|D] [ES|W] [E|TT] [B|E] [WE|RB]
[B|UN] [D|E] [S|W] [E|TT] [BE|W] [ER|B]
[B|U] [N|D] [ES|W] [E|TT] [BE|W] [E|RB]
[B|UN] [D|E] [S|W] [E|TT] [BE|W] [E|RB]
[B|U] [N|D] [ES|W] [E|TT] [B|EW] [ER|B]
[B|UN] [DE|SW] [E|TT] [BE|W] [E|RB]
[B|U] [N|D] [E|S] [WE|TT] [B|EW] [E|RB]
[B|U] [N|D] [E|S] [WE|TT] [B|E] [WER|B]
[B|U] [N|D] [E|S] [WE|TT] [B|EW] [ER|B]
[B|U] [N|D] [E|S] [WE|TT] [B|E] [WE|RB]
[B|UN] [DE|SW] [E|TT] [B|EW] [ER|B]
[B|UN] [DE|SW] [E|TT] [B|E] [WER|B]
[B|U] [ND|ES] [WE|TT] [B|E] [WER|B]
[B|UN] [DE|SW] [E|TT] [B|EW] [E|RB]
[B|U] [ND|ES] [WE|TT] [B|EW] [ER|B]
[B|UN] [DE|SW] [E|TT] [B|E] [WE|RB]
[B|U] [N|DE] [S|W] [E|TT] [B|E] [WER|B]
[B|U] [N|DE] [S|W] [E|TT] [B|EW] [ER|B]
[B|U] [N|D] [ES|W] [E|TT] [B|E] [W|E] [R|B]
[B|U] [N|D] [E|SW] [E|TT] [BE|W] [ER|B]
[B|UN] [DE|S] [WE|TT] [B|E] [WE|RB]
[B|U] [ND|ES] [WE|TT] [B|E] [WE|RB]
[B|UN] [DE|SW] [E|TT] [B|E] [W|E] [R|B]
[B|U] [N|D] [E|SW] [E|TT] [BE|WE] [R|B]
[B|U] [N|D] [ES|W] [E|TT] [B|E] [WER|B]
[B|U] [N|DE] [S|W] [E|TT] [B|E] [WE|RB]
[B|U] [N|DE] [S|W] [E|TT] [B|EW] [E|RB]
[B|U] [ND|ES] [WE|TT] [B|EW] [E|RB]
[B|UN] [DE|S] [WE|TT] [BE|W] [ER|B]
```

```
[B|UN] [DE|S] [WE|TT] [BE|WE] [R|B]
[B|UN] [D|E] [S|W] [E|TT] [BE|WE] [R|B]
[B|UN] [DE|SW] [E|TT] [BE|W] [ER|B]
[B|UN] [DE|S] [WE|TT] [B|E] [W|E] [R|B]
[B|U] [ND|E] [S|W] [E|TT] [BE|W] [ER|B]
[B|UN] [DE|S] [WE|TT] [BE|W] [E|RB]
[B|U] [N|D] [E|SW] [E|TT] [B|EW] [ER|B]
[B|U] [N|D] [E|SW] [E|TT] [B|E] [WER|B]
[B|U] [ND|E] [S|W] [E|TT] [BE|WE] [R|B]
[B|U] [ND|E] [S|W] [E|TT] [BE|W] [E|RB]
[B|U] [N|D] [ES|W] [E|TT] [BE|W] [ER|B]
[B|U] [N|DE] [S|W] [E|TT] [B|E] [W|E] [R|B]
[B|UN] [DE|SW] [E|TT] [BE|WE] [R|B]
[B|U] [N|D] [E|S] [WE|TT] [BE|W] [E|RB]
[B|UN] [DE|S] [WE|TT] [B|E] [WER|B]
[B|U] [ND|ES] [WE|TT] [B|E] [W|E] [R|B]
[B|U] [N|D] [E|S] [WE|TT] [B|E] [W|E] [R|B]
[B|U] [N|D] [ES|W] [E|TT] [BE|WE] [R|B]
[B|U] [N|D] [E|SW] [E|TT] [B|E] [WE|RB]
[B|U] [N|DE] [S|W] [E|TT] [BE|W] [E|RB]
[B|U] [N|D] [E|SW] [E|TT] [B|EW] [E|RB]
[B|U] [ND|ES] [WE|TT] [BE|W] [E|RB]
[B|U] [ND|E] [S|W] [E|TT] [B|E] [WE|RB]
[B|U] [ND|E] [S|W] [E|TT] [B|EW] [E|RB]
[B|U] [ND|E] [S|W] [E|TT] [B|E] [W|E] [R|B]
[B|U] [ND|E] [S|W] [E|TT] [B|E] [WER|B]
[B|U] [ND|E] [S|W] [E|TT] [B|EW] [ER|B]
[B|U] [N|D] [E|S] [WE|TT] [BE|W] [ER|B]
[B|U] [N|D] [E|S] [WE|TT] [BE|WE] [R|B]
[B|U] [N|D] [E|SW] [E|TT] [B|E] [W|E] [R|B]
[B|UN] [DE|S] [WE|TT] [B|EW] [E|RB]
[B|U] [ND|ES] [WE|TT] [BE|W] [ER|B]
[B|UN] [DE|S] [WE|TT] [B|EW] [ER|B]
[B|U] [ND|ES] [WE|TT] [BE|WE] [R|B]
[B|U] [N|DE] [S|W] [E|TT] [BE|W] [ER|B]
[B|U] [N|DE] [S|W] [E|TT] [BE|WE] [R|B]
Das Wort kann auf einem Nummernschild stehen.

Möglichkeiten für das Wort 'CLINTON':
[C|LI] [NT|ON]
Das Wort kann auf einem Nummernschild stehen.

Möglichkeiten für das Wort 'INFORMATIK':
[IN|FO] [R|M] [AT|IK]
Das Wort kann auf einem Nummernschild stehen.

Möglichkeiten für das Wort 'ETHERNET':
[E|T] [HE|RN] [E|T]
[E|T] [HE|R] [NE|T]
[E|T] [HE|R] [N|ET]
[E|TH] [E|RN] [E|T]
[E|TH] [E|R] [N|ET]
[E|TH] [E|R] [NE|T]
[E|T] [H|E] [RN|ET]
[E|T] [H|ER] [N|ET]
[E|T] [HER|N] [E|T]
[E|TH] [ER|N] [E|T]
[E|T] [H|E] [R|N] [E|T]
```

```
[E|T] [H|ER] [NE|T]
Das Wort kann auf einem Nummernschild stehen.

Möglichkeiten für das Wort
'LLANFAIRPWLLGWYNGYLLGOGERYCHWYRNDROBWLLLLANTYSILIOGOGOGOCH':

Das Wort kann nicht auf einem Nummernschild stehen.
Möglichkeiten für das Wort
'RINDFLEISCHETIKETTIERUNGSÃŒBERWACHUNGSAUFGABENÃŒBERTRAGUNGSGESETZ':

Das Wort kann nicht auf einem Nummernschild stehen.

Möglichkeiten für das Wort 'SOFTWARE':
[S|O] [FT|W] [A|RE]
[SO|FT] [WA|RE]
[SO|FT] [W|A] [R|E]
[S|O] [F|TW] [A|RE]
[S|O] [FT|WA] [R|E]
[S|O] [F|T] [WA|RE]
[S|O] [F|T] [W|A] [R|E]
Das Wort kann auf einem Nummernschild stehen.
```

Wie wir sehen, ist die Anzahl der Möglichkeiten sehr verschieden.

Quellcode

Unwichtige Teile des Programms müssen hier nicht abgedruckt werden.

Die Klasse Autoscrabble (main-class):

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class Autoscrabble {
	static ArrayList<String> kuerzelliste = new ArrayList<String>();
	static Scanner input = new Scanner(System.in);
	static ArrayList<String> loesung = new ArrayList<String>();
	static ArrayList<String> wörter = new ArrayList<String>();

	public static void main(String[] args) throws FileNotFoundException {
		// Umfüllen der Kürzel aus der Kürzeldatei in eine Liste -> besserer
		// Zugang
		try {
			File kuerzel = new File("E:\\workspace_desktop\\kuerzelliste.txt");
			Scanner scanner = new Scanner(kuerzel);
			while (scanner.hasNextLine()) {
				kuerzelliste.add(scanner.nextLine().toUpperCase());
			}
			scanner.close();
			System.out.println(kuerzelliste);
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
		// Umfüllen der Wörter von der BWInf-Website in eine Liste -> besserer
		// Zugang
		try {
			File woerter = new File("E:\\workspace_desktop\\autoscrabble.txt");
			Scanner scanner = new Scanner(woerter);
			while (scanner.hasNextLine()) {
				wörter.add(scanner.nextLine().toUpperCase());
			}
			scanner.close();
```

```
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                }
                System.out.println("Welches Wort?('0' für eigenes, '1' für Wort aus Liste)");
                String name;
                if (input.nextInt() == 0) {
                        System.out.print("Bitte Wort eingeben: ");
                        name = input.next().toUpperCase();
                } else {
                        // Auswählen welches Wort untersucht werden soll
                        System.out.print("Welches Wort?( Zahl von 0 bis " + (wörter.size() - 1) + ") :
");
                        int pos = input.nextInt();
                        name = wörter.get(pos);
                }
                System.out.println("Möglichkeiten für das Wort '" + name + "':");
                String kuezel = "";
                // Anfangs drei Möglichkeiten für das Kürzel des ersten Nummernschildes:
                // Bestehend aus den ersten 1, 2 oder 3 Buchstaben des Worts
                // Prüfen welche dieser Nummernschildanfänge möglich sind, pro
                // Möglichkeit ein neuer Thread, der ab der Möglichkeit weitere
                // Möglichkeiten prüft( siehe Abzweigungen)
                if (name.length() >= 1) {// erster Buchstabe
                        kuezel = (name.charAt(0) + "");
                        if (kuerzelliste.contains(kuezel)) {
                                loesung.add(kuezel);
                                Thread x = new Thread(new Abzweigung(loesung, 1, name));
                                x.start();
                                loesung.clear();
                        }
                }
                if (name.length() >= 2) {// erste zwei Buchstaben
                        kuezel = (name.charAt(0) + "" + name.charAt(1) + "");
                        if (kuerzelliste.contains(kuezel)) {
                                loesung.add(kuezel);
                                Thread y = new Thread(new Abzweigung(loesung, 2, name));
                                y.start();
                                loesung.clear();
                        }
                }
                if (name.length() >= 3) {// erste drei Buchstaben
                        kuezel = (name.charAt(0) + "" + name.charAt(1) + "" + name.charAt(2) + "");
                        if (kuerzelliste.contains(kuezel)) {
                                loesung.add(kuezel);
                                Thread z = new Thread(new Abzweigung(loesung, 3, name));
                                z.start();
                                loesung.clear();
                        }
                }
        }
}
```

Die Klasse Abzweigung:

```
import java.util.ArrayList;

public class Abzweigung implements Runnable {
        ArrayList<String> loesung = new ArrayList<String>();
        int pos;
        String name;
        String kuezel = "";

        public Abzweigung(ArrayList<String> list, int position, String Name) {
                // Speichern der Eingabeparameter
                for (String s : list) {
                        loesung.add(s);
                }
                pos = position;
                name = Name;
        }

        @Override
```

```
public void run() {
        // Prüfen, ob schon am Wortende angelangt: wenn die Position nurnoch ein
        // oder zwei Buchstaben vom Wortende entfernt ist -> diese müssen kein
        // Kürzel sein, da das Wortende der Mittelteil des Nummernschilds ist,
        // der ein oder zwei Buchstaben lang sein kann und dessen Buchstaben
        // egal sind
        if (name.length() == pos + 1) {// letzter Mittelteil ein Buchstabe lang
                kuezel = (name.charAt(pos) + "");
                loesung.add(kuezel);
                print();// Wenn am Ende angelangt, Loesung ausgeben
                loesung.remove(loesung.size() - 1);
        }
        if (name.length() == pos + 2) {// letzter Mittelteil zwei Buchstaben
                                                                // lang
                kuezel = (name.charAt(pos) + "" + name.charAt(pos + 1) + "");
                loesung.add(kuezel);
                print();// Wenn am Ende angelangt, Loesung ausgeben
                loesung.remove(loesung.size() - 1);
        }
        // Jede new Abzweigung kann sechs Möglichkeiten haben, die in einer
        // neuen new Abzweigung untersucht werden:
        // Mittelteil 1 lang + Kürzel des nächsten Nummernschilds 1 lang
        // Mittelteil 1 lang + Kürzel des nächsten Nummernschilds 2 lang
        // Mittelteil 1 lang + Kürzel des nächsten Nummernschilds 3 lang
        // Mittelteil 2 lang + Kürzel des nächsten Nummernschilds 1 lang
        // Mittelteil 2 lang + Kürzel des nächsten Nummernschilds 2 lang
        // Mittelteil 2 lang + Kürzel des nächsten Nummernschilds 3 lang
        if (name.length() >= pos + 2) {
                // Mittelteil 1 lang + Kürzel des nächsten Nummernschilds 1 lang
                kuezel = (name.charAt(pos) + "");
                loesung.add(kuezel);
                kuezel = (name.charAt(pos + 1) + "");
                if (Autoscrabble.kuerzelliste.contains(kuezel)) {
                        loesung.add(kuezel);
                        // Wenn Möglichkeit möglich( Kürzel in Kürzelliste enthalten)
                        // neue Abzweigung
                        neuerThread(loesung, pos + 2, name);
                        loesung.remove(loesung.size() - 1);
                }
                loesung.remove(loesung.size() - 1);
        }
        if (name.length() >= pos + 3) {
                // Mittelteil 1 lang + Kürzel des nächsten Nummernschilds 2 lang
                kuezel = (name.charAt(pos) + "");
                loesung.add(kuezel);
                kuezel = (name.charAt(pos + 1) + "" + name.charAt(pos + 2) + "");
                if (Autoscrabble.kuerzelliste.contains(kuezel)) {
                        loesung.add(kuezel);
                        // Wenn Möglichkeit möglich( Kürzel in Kürzelliste enthalten)
                        // neue Abzweigung
                        neuerThread(loesung, pos + 3, name);
                        loesung.remove(loesung.size() - 1);
                }
                loesung.remove(loesung.size() - 1);
        }
        if (name.length() >= pos + 4) {
                // Mittelteil 1 lang + Kürzel des nächsten Nummernschilds 3 lang
                kuezel = (name.charAt(pos) + "");
                loesung.add(kuezel);
                kuezel = (name.charAt(pos + 1) + "" +
                                name.charAt(pos + 2) + "" + name.charAt(pos + 3) + "");
                if (Autoscrabble.kuerzelliste.contains(kuezel)) {
                        loesung.add(kuezel);
                        // Wenn Möglichkeit möglich( Kürzel in Kürzelliste enthalten)
                        // neue Abzweigung
                        neuerThread(loesung, pos + 4, name);
                        loesung.remove(loesung.size() - 1);
                }
                loesung.remove(loesung.size() - 1);
        }
        if (name.length() >= pos + 3) {
                // Mittelteil 2 lang + Kürzel des nächsten Nummernschilds 1 lang
```

```
                kuezel = (name.charAt(pos) + "" + name.charAt(pos + 1) + "");
                loesung.add(kuezel);
                // Wenn Möglichkeit möglich( Kürzel in Kürzelliste enthalten) neue
                // Abzweigung
                kuezel = (name.charAt(pos + 2) + "");
                if (Autoscrabble.kuerzelliste.contains(kuezel)) {
                        loesung.add(kuezel);
                        neuerThread(loesung, pos + 3, name);
                        loesung.remove(loesung.size() - 1);
                }
                loesung.remove(loesung.size() - 1);
        }
        if (name.length() >= pos + 4) {
                // Mittelteil 2 lang + Kürzel des nächsten Nummernschilds 2 lang
                kuezel = (name.charAt(pos) + "" + name.charAt(pos + 1) + "");
                loesung.add(kuezel);
                // Wenn Möglichkeit möglich( Kürzel in Kürzelliste enthalten) neue
                // Abzweigung
                kuezel = (name.charAt(pos + 2) + "" + name.charAt(pos + 3) + "");
                if (Autoscrabble.kuerzelliste.contains(kuezel)) {
                        loesung.add(kuezel);
                        neuerThread(loesung, pos + 4, name);
                        loesung.remove(loesung.size() - 1);
                }
                loesung.remove(loesung.size() - 1);
        }
        if (name.length() >= pos + 5) {
                // Mittelteil 2 lang + Kürzel des nächsten Nummernschilds 3 lang
                kuezel = (name.charAt(pos) + "" + name.charAt(pos + 1) + "");
                loesung.add(kuezel);
                // Wenn Möglichkeit möglich( Kürzel in Kürzelliste enthalten) neue
                // Abzweigung
                kuezel = (name.charAt(pos + 2) + "" +
                                name.charAt(pos + 3) + "" + name.charAt(pos + 4) + "");
                if (Autoscrabble.kuerzelliste.contains(kuezel)) {
                        loesung.add(kuezel);
                        neuerThread(loesung, pos + 5, name);
                        loesung.remove(loesung.size() - 1);
                }
                loesung.remove(loesung.size() - 1);
        }
}

public void warten(long millis) {
        // Methode für Thread.sleep(), um den Code zu verkürzen
        try {
                Thread.sleep(millis);
        } catch (InterruptedException e) {
        }
}

// Da es bei langen Wörtern sehr viele Möglichkeiten gibt und dadurch sehr
// viele Threads benötigt werden, kommt es sehr oft zu OutOfMemoryErrors
// Diese werden durch diese Methode vermieden, indem der Thread erst kreiert
// wird, wenn es genug Speicherplatz gibt
public void neuerThread(ArrayList<String> loesung, int pos, String name) {
        boolean mem;
        do {
                try {
                        // Thread wird kreiert
                        Thread x = new Thread(new Abzweigung(loesung, pos, name));
                        x.start();
                        mem = false;
                } catch (OutOfMemoryError x) {
                        mem = true;
                        warten(1);
                        // Bei OutOfMemoryErrors wird nach einer Millisekunde der
                        // Prozess wiederholt
                }
        } while (mem);
}
```

```
        public void print() {
                // Ausgeben der Möglichkeit in Zweierpärchen:
                // [ erster Teil | zweiter Teil ]
                String str = "";
                for (int x = 0; x < loesung.size(); x++) {
                        if (x % 2 == 0) {
                                str += ("[" + loesung.get(x) + "|");
                        } else {
                                str += (loesung.get(x) + "] ");
                        }
                }
                System.out.println(str);
        }
}
```