Aufgabe 3: Wortsuche

Team-ID: 00921

Team: Ctrl + Intelligence

Bearbeiter dieser Aufgabe:
Tim Eismar

21. November 2021

Inhalt

Lösungsidee

Die Idee ist, dass eine Matrix mit gegebener Größe erstellt wird. Danach wird versucht, die Wörter unter gegebenen Regeln (Diese verändern sich durch die Schwierigkeits-Stufen) in die Matrix einzufügen. Hierbei wird dem ersten Buchstaben des Wortes eine zufällige Koordinate der Matrix zugewiesen. Abschließend werden die freien Felder der Matrix mit einer Liste an Buchstaben gefüllt.

Schwierigkeits-Stufen / Regeln

Stufe	*Ausrichtung*	*Wort*	*Füll-Liste*
Stufe 1	Horizontal	Normal	Alphabet
Stufe 2	Vertikal	Normal	Alphabet
Stufe 3	Horizontal und Vertikal	Normal	Alphabet
Stufe 4	Horizontal und Vertikal	Manchmal Rückwärts	Alphabet
Stufe 5	Horizontal und Vertikal	Manchmal Rückwärts	Buchstaben der gesuchten Wörter

Umsetzung

User-Interface

Am Anfang wird eine JFileChooser erstellt, um die gegebene Beispieldatei auszuwählen. Danach wird der Nutzer, per Konsole, nach seinem gewünschten Schwierigkeitsgrad gefragt. Mit diesen beiden Infos wird dann eine Wortmatrix erstellt. Diese wird dann samt einer Liste der gesuchten Wörter, in der Konsole, ausgegeben.

Intern

Die Matrix

Die „Matrix“ wird durch ein zweidimensionales char-Array realisiert, welches die aus der Datei gelesene Größe hat.

Datei auslesen

Die Methode *readLine* nutzt die API „java.nio“, um eine gewünschte Zeile des Textdokuments als String wiederzugeben.

Der „Füll“-Algorithmus

Die Wörter werden in einem String-Array übergeben. Beim „Füllen“ des Arrays, geht die Methode *fillMat* mit einer for-Schleife durch das String-Array und bearbeitet somit die Wörter nacheinander. Zum „Füllen“ nutzt sie mehrere try/catch-Statements. In diesen wird für ein Wort eine zufällige Koordinate im Array ausgewählt. Von dieser ausgehend wird dann entsprechend der Regeln versucht, das aktuelle Wort in die Matrix einzufügen. Hierbei wird allerdings erst versucht ein String aus Platzhaltersymbolen (#) mit der Länge des Wortes einzufügen, wenn dies funktioniert, wird dann das Wort eingefügt. Dieser Schritt ist notwendig da die Matrix sonst von Fragmenten aus fehlgeschlagenen Versuchen gefüllt wird. Denn durch die try/catch-Statemenst kann es passieren,

dass ein Wort zur Hälfte eingefügt ist, dann festgestellt wird, dass man z.B. outOfBounds ist. Hier wird die Hälfte des Wortes allerdings nicht automatisch entfernt, sondern bleibt im Array, deswegen die Vorstufe mit dem Platzhaltersymbol. Wenn das String-Array mit den Wörtern dann durchgelaufen ist, werden die restlichen Felder des Arrays mit zufälligen Buchstaben gefüllt, diese stammen den Regeln entsprechend entweder aus einem String, der das gesamte Alphabet enthält, oder einem String mit allen Buchstaben, die in den zu suchenden Wörtern vorkommen.

Die Schwierigkeits-Stufen / Regeln

Um auf die Auswahl der Schwierigkeitsstufe zu reagieren, werden in und um die try/catch-Statements mehrere switch-Statements genutzt. So wird beispielsweise in einem switch-Statement entschieden ob Wörter rückwärts in die Matrix geschrieben werden.

Die Bedingung für einen Neustart

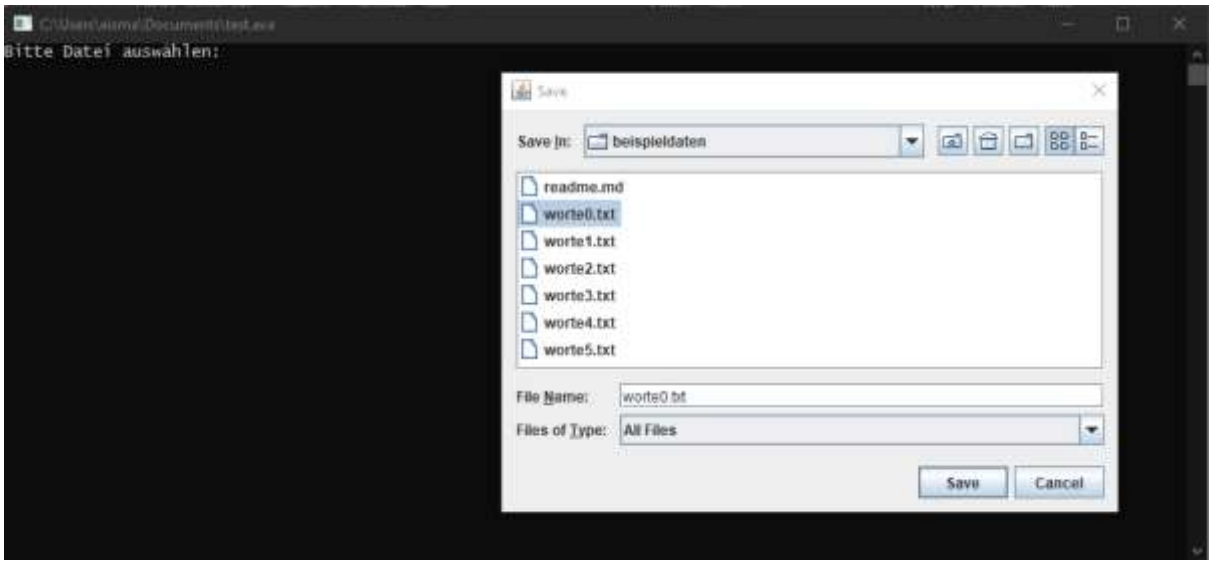
Da es durch den Zufallsfaktor passieren kann, dass ein, bzw. mehrere Wörter, so platziert wurden, dass für die übrigen nicht mehr genug Platz vorhanden ist, gibt es eine Neustartbedingung. Diese funktioniert, indem sie vor jedem neuen Buchstaben prüft, wie viel Zeit seit dem Start des „Füll“-Algorithmus vergangen ist, wenn diese mehr als 0,5 Sekunden beträgt, wird der Algorithmus mit einer leeren Matrix neugestartet. Die 0,5 Sekunden haben sich bei den Tests als genügend Zeit herausgestellt, können aber aufgrund von leistungsunterschieden zwischen PCs, am Anfang des Programms angepasst werden.

StringBuilder

Die Library „StringBuilder“ wird, in dem Programm, zum einen genutzt, um einen String umzudrehen und zum anderen um ein String-Array von Wörtern zu einem String mit allen Buchstaben die vorkommen zu machen.

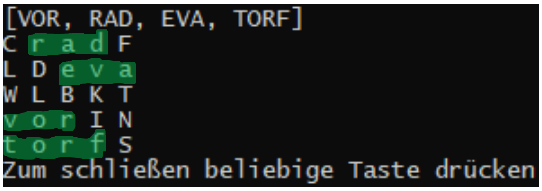
Beispiele

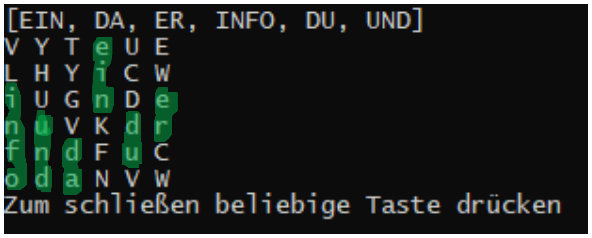
JFileChooser

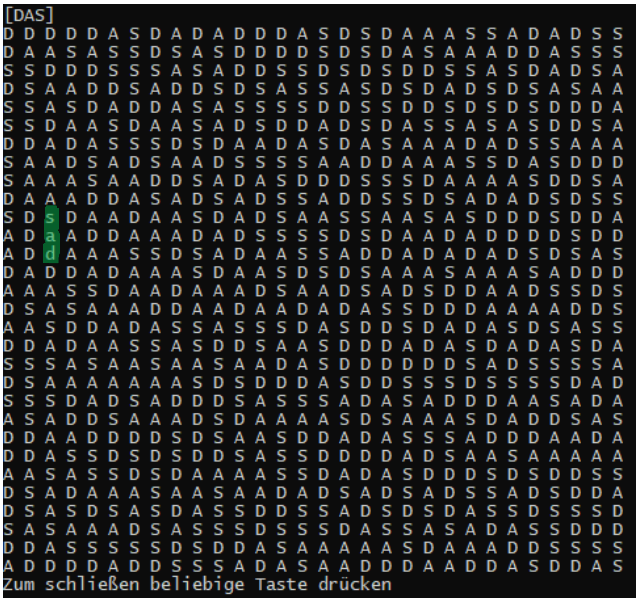


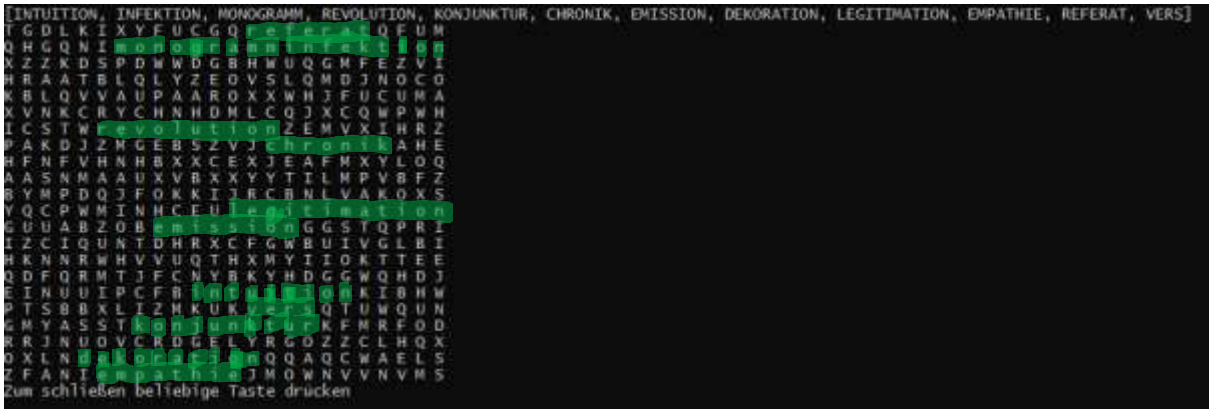
Ausgabe

Zur besseren Sichtbarkeit sind hier die Lösungswörter in Kleinbuchstaben geschrieben, dies ist im beigefügten Programm nicht der Fall.









Hier bewusst nur horizontal, um das Suchen der Wörter zu vereinfachen

Quellcode

Algorithmus für das senkrechte Schreiben

```
case 1: { //Senkrecht

                    boolean checker = false;


                    while (checker == false) {
                        try {
                            for (int t = 0; t < word[i].length(); t++) {

                                if (System.currentTimeMillis() - start > end) { // Abbruch nach X-Sekunden
                                    return fillMat(fileP, word, matrix, difC, end);
                                }
                                if (mat[randomX + t][randomY] == '\u0000' || mat[randomX + t][randomY] ==
"#".charAt(0)) { //Es wird getestet ob das wort reinpassst
                                    mat[randomX + t][randomY] = "#".charAt(0); //Platzhalter
                                } else {
                                    randomX = random.nextInt(mat.length); //Neue Startkoordinaten
                                    randomY = random.nextInt(mat[randomX].length);
                                }

                            }
                            for (int t = 0; t < word[i].length(); t++) {
                                if (System.currentTimeMillis() - start > end) {
                                    return fillMat(fileP, word, matrix, difC, end);
                                }
                                if (mat[randomX + t][randomY] == '\u0000' || mat[randomX + t][randomY] ==
"#".charAt(0)) { //Wort wird eingefügt
                                    mat[randomX + t][randomY] = word[i].toLowerCase().charAt(t);
                                } else {
                                    randomX = random.nextInt(mat.length);//Absicherung
                                    randomY = random.nextInt(mat[randomX].length);
                                }

                            }
                            checker = true;
                        } catch (Exception e) {
                            try {
                                for (int t = 0; t < word[i].length(); t++) {
                                    if (System.currentTimeMillis() - start > end) {
                                        return fillMat(fileP, word, matrix, difC, end);
                                    }
                                    if (mat[randomX + t][randomY] == '\u0000' || mat[randomX + t][randomY
] == "#".charAt(0)) {
                                        mat[randomX + t][randomY] = "#".charAt(0);
                                    } else {
                                        randomX = random.nextInt(mat.length);
                                        randomY = random.nextInt(mat[randomX].length);
                                    }

                                }
                                for (int t = 0; t < word[i].length(); t++) {
                                    if (System.currentTimeMillis() - start > end) {
                                        return fillMat(fileP, word, matrix, difC, end);
                                    }
                                    if (mat[randomX + t][randomY] == '\u0000' || mat[randomX + t][randomY
] == "#".charAt(0)) {
                                        mat[randomX + t][randomY] = word[i].toLowerCase().charAt(t);
                                    } else {
                                        randomX = random.nextInt(mat.length);
                                        randomY = random.nextInt(mat[randomX].length);
                                    }

                                }
                                checker = true;
                            } catch (Exception g) {
                                if (randomX != 0) {
                                    randomX--;
                                }
                            }
                        }
                    }
                }
                break;
```

Hilfsmethoden

```
public static String readLine(String fileLoc, int lin) throws IOException {
//Hilfsmethode zum Dateilesen
        String line = Files.readAllLines(Paths.get(fileLoc)).get(lin);
        return line;
    }

public static String toLetterArr(String[] words) {//WortArray zum Buchstabenstring
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < words.length; i++) {
            stringBuilder.append(words[i]);
        }
        String finalString = stringBuilder.toString();
        //System.out.println(finalString);
        return finalString;
    }

public static String[] wordArr(String fileLoc) throws IOException {
//Wörterrray wird erstellt
        String size = readLine(fileLoc, 1);
        String[] sp = new String[Integer.valueOf(size)];
        int e = 2;
        for (int i = 0; i < Integer.valueOf(size); i++) {
            sp[i] = readLine(fileLoc, e);
            e++;
        }

        return sp;
    }
```