Aufgabe 1: Störung

Team-ID: 00225

Team-Name: Simon und Benjamins Teams

Bearbeiter dieser Aufgabe:
Dominik Luckert

13. November 2022

Inhaltsverzeichnis

1 Lösungsidee

Als ersten Schritt sucht man das erste Wort der gesuchten Stelle im Buchtext. Wenn das erste Wort mit keinem Wort im Buchtext übereinstimmt, kann es keine Lösung geben.

Anschließend wird von dieser Stelle aus geprüft, ob der Rest der gesuchten Stelle mit der gleich grossen Strecke im Buchtex übereinstimmt. Wenn die Stellen nicht gleich sind, weil zum Beispiel kommende Wörter im Buchtext anders sind als in der gesuchten Stelle oder die Anzahl der unbekannten Wörter (dargestellt mit _) nicht mit dem Buchtext übereinstimmt, wird die Lösung verworfen. Wenn eine passende Stelle im Buch gefunden wurde, wird diese dem Benutzer ausgegeben und weitere mögliche Stellen werden nach dem gleichen Prinzip im Buchtext gesucht.

2 Umsetzung

Die Idee wurde in Java umgesetzt. Buchtext und die gesuchte Stelle werden zunächst aus der Datei gelesen und dann in zwei Arrays geladen. In den Arrays hat jedes Wort einen eigenen Index. Für das Einlesen werden Methoden einer `Scanner` Umgebung genutzt, welche mit `import java.util.Scanner;` zur Verfügung gestellt werden. Damit die Wörter aus Buchtext und der gesuchten Stelle miteinander verglichen werden können, müssen mehrere Vorkehrungen getroffen werden:

- Da in den zu testenden Dateien die Buchstallen ohne Satzzeichen stehen: Die Methode `void deleteMarks()` entfernt alle potenziell auftretenden Satzzeichen wie Punkt, Komma oder Anführungszeichen. Hierfür wird die String-Methode `import java.util.Scanner;replace(...)` wie folgt verwendet:
  `betroffenesWort.replace(zuLoeschendesSatzzeichen, "")`.

  Bemerkung: Die Methode stellt sicher, dass auch Wörter, an denen mehrere Satzzeichen stehen, vollständig davon gelöst werden. Die entfernten Satzeichen in der Methode wurden nach den entsprechend verwendeten Zeichen im Buchtext angepasst und sind einfach erweiterbar.

Alle Beispielstellen enthalten einhaltlich keine Satzzeichen. Dennoch würden aus Versehen auftretende Kommata und Punkte entfernt werden. Bei Bedarf könnte man dies schnell auf mehr Satzzeichen ausweiten.

- Das Array `int[] achar;` speichert, für den einzulesenden Text, ob am Index `i` ein Wort oder ein beliebiges Wort(dargestellt mit _) steht.

Zusätzlich muss bei allen Vergleichen von Buchtext und gesuchter Stelle die Groß-und Kleinschreibung missachtet werden, da alle Wörter aus den gesuchten Stellen kleingeschrieben wurden. Damit die Wörter ohne grossen Aufwand trotzdem miteinander verglichen werden können, wird die bereits existierende String -Methode mit `wort1.equalsIgnoreCase(wort2)` verwendet.

Nachdem alle beschriebenen Vorkehrungen getroffen wurden, funktioniert das tatsächliche Vergleichen wie folgt:

- Die Methode `void findStart()` sucht nach Textstellen, für die ein Wort im Buch mit dem ersten Wort in der gesuchten Stelle übereinstimmt. Von jeder gefundenen Stelle wird der Index der Methode `void checking(...)` übergeben, die den Rest der gesuchten Datei mit dem Buchtext überprüft. Wenn kein Wort im Buch mit dem ersten Wort der Datei übereinstimmt, ist keine mögliche Stelle im Buchtext enthalten und es wird dem Benutzer eine entsprechende Meldung ausgegeben.

  Bemerkung: Wenn die gesuchte Stelle mit einem _ beginnt, wird nicht gesucht (alle Beispieldateien starten mit einem Wort).

- Anschließend wird in der Methode `void checking(...)` überprüft, ob die gesamte gesuchte Stelle mit der gefundenen Stelle im Buch übereinstimmt. Damit dies zutrifft, müssen entweder die Wörter an der gleichen Stelle (dem entsprechenden Index im Array) übereinstimmen, oder in der gesuchten Stelle steht ein "_", also das Wort darf beliebig sein. Da das Array, welches die Wörter der gesuchten Stelle speichert, viel kleiner ist als der gesamte Buchtext, stimmen die Indizees beim Vergleichen logischerweise nicht überein. Zum Beispiel ist das erste Wort der gesuchten Stelle am Index 0 "Hallo". Im Buchtext kommt das Wort aber erst am Index x!=0 vor. Aus diesem Grund wird der Methode `void checking(...)` der Index `j` übergeben, an welchem das Wort im Buchtext gefunden wurde. Der Index `i` an der gesuchten Stelle ist folglich der Index `i+j` im Buchtext. So kann verglichen werden.

  Wenn eine übereinstimmende Textstelle sprich eine Lösung für die gesuchte Stelle gefunden wurde, wird diese ausgegeben und es wird nach weiteren gesucht.

3 Beispiele

Wie bereits angedeutet fällt auf, dass die Wörter der gesuchten Stelle nur kleingeschrieben sind, während der gefundene Buchtext nach deutscher Rechtschreibung gross-und kleingeschrieben ist.

Beispiele 1 und 2 liefern mehr als eine Lösung.

```
Ausgewählte Lücke im Text: das _ mir _ _ _ vor
-------------------------------------------------------
Lösung gefunden!: Das kommt mir gar nicht richtig vor
```

Abbildung 1: Beispiel 0

```
Ausgewählte Lücke im Text: ich muß _ clara _
-------------------------------------------------------
Lösung gefunden!: Ich muß in Clara verwandelt
Lösung gefunden!: Ich muß doch Clara sein
```

Abbildung 2: Beispiel 1

```
Ausgewählte Lücke im Text: fressen _ gern _
-------------------------------------------------------
Lösung gefunden!: Fressen Katzen gern Spatzen
Lösung gefunden!: Fressen Katzen gern Spatzen
Lösung gefunden!: Fressen Spatzen gern Katzen
```

Abbildung 3: Beispiel 2

```
Ausgewählte Lücke im Text: das _ fing _
-------------------------------------------------------
Lösung gefunden!: das Spiel fing an
```

Abbildung 4: Beispiel 3

```
Ausgewählte Lücke im Text: ein _ _ tag
-------------------------------------------------------
Lösung gefunden!: ein sehr schöner Tag
```

Abbildung 5: Beispiel 4

```
Ausgewählte Lücke im Text: wollen _ so _ sein
-------------------------------------------------------
Lösung gefunden!: Wollen Sie so gut sein
```

Abbildung 6: Beispiel 5

3.1 eigenes Beispiel

Das erste Beispiel ist eine gesuchte Stelle, welche kein bestimmtes Anfangswort enthält. Hier ist keine Lösung möglich:

```
Ausgewählte Lücke im Text: _
-------------------------------------------------------
Keine Lösung möglich
```

Abbildung 7: eigenes Beispiel 1

Das zweite Beispiel zeigt, dass auch mehrere Lösungen (wenn vorhanden) ausgegeben werden:

```
Ausgewählte Lücke im Text: Alice und _
-------------------------------------------------------
Lösung gefunden!: Alice und um
Lösung gefunden!: Alice und wenn
Lösung gefunden!: Alice und Thränen
Lösung gefunden!: Alice und sie
Lösung gefunden!: Alice und sah
Lösung gefunden!: Alice und nachdem
Lösung gefunden!: Alice Und wo
Lösung gefunden!: Alice und sprach
Lösung gefunden!: Alice und es
Lösung gefunden!: Alice und damit
Lösung gefunden!: Alice und diese
Lösung gefunden!: Alice und sie
Lösung gefunden!: Alice und ihre
```

Abbildung 8: eigenes Beispiel 2

4 Quellcode

```
1
import java.io.*;
3 import java.util.Scanner;

5 public class Suche{

7   private static String gesucht = new String();
    private static String text;  //speichert kompletten Buchtext
9   private static String[] atext; //speichert jedes Wort des Buchtext in Reihenfolge
    private static String[] agesucht; //speichert jedes Wort des gesuchten Textes
11  private static int[] achar;        //0=_ / 1=word

13  public static void main(String[]args) throws IOException{

15    //*************Hier File einlesen***********************************************
      File file = new File("stoerung5.txt");
17    //******************************************************************************

19    Scanner sc = new Scanner(file, "UTF-8");
      gesucht = sc.nextLine(); //liest gesuchte Textstelle ein
21
      System.out.println("Ausgewaehlte␣Luecke␣im␣Text:␣" + gesucht);
23    System.out.println("-------------------------------------------------------");

25    reading();     //step 1: liest Buchtext ein
    }
27
    public static void reading() throws IOException{
29
      File file = new File("Alice_Im_Wunderland.txt");
31    Scanner sc = new Scanner(file, "UTF-8");

33    while (sc.hasNextLine()) {
        text += sc.nextLine();
35      text += "␣";
      }
37
      preperation(); //step 2: initialisiert arrays
39  }


41
    public static void preperation(){
43
      atext = text.split("␣");
45
      agesucht = gesucht.split("␣");
47
      achar = new int[agesucht.length];
49
      for(int i = 0; i < achar.length; i++){
51      if(!agesucht[i].equals("_")){
          achar[i] = 1;
53      } else {
          achar[i] = 0;
55      }
      }
57
      deleteMarks();  //step 3: loescht satzzeichen im buchtext
59
    }
61
    public static void deleteMarks(){
63    for (int i = 0; i < atext.length; i++) {
        if (atext[i].contains(",")) {
65        atext[i] = atext[i].replace(",","");
        }
67      if (atext[i].contains(".")) {
          atext[i] = atext[i].replace(".","");
69      }
        if (atext[i].contains("""))␣{
71 ␣␣␣␣␣␣␣␣atext[i]␣=␣atext[i].replace("""","");
```

```
        }
73      if(atext[i].contains("""")){
␣␣␣␣␣␣␣␣atext[i]␣=␣atext[i].replace("""","");
75      }
        if (atext[i].contains(";")) {
77        atext[i] = atext[i].replace(";","");
        }
79      if (atext[i].contains("?")) {
          atext[i] = atext[i].replace("?","");
81      }
        if (atext[i].contains("(")) {
83        atext[i] = atext[i].replace("(","");
        }
85      if (atext[i].contains(")")) {
          atext[i] = atext[i].replace(")","");
87      }
        if (atext[i].contains("_")) {
89        atext[i] = atext[i].replace("_","");
        }
91      if (atext[i].contains(":")) {
          atext[i] = atext[i].replace(":","");
93      }
        if (atext[i].contains("!")) {
95        atext[i] = atext[i].replace("!","");
        }
97    }
     for (int i = 0; i < agesucht.length; i++) {
99      if (agesucht[i].contains(",")) {
          agesucht[i] = agesucht[i].replace(",","");
101     }
        if (agesucht[i].contains(".")) {
103       agesucht[i] = agesucht[i].replace(".","");
        }
105   }

107   findStart();
    }
109
    public static void findStart(){
111   int count = 0;
      for(int i = 0; i < atext.length; i++){
113     if (atext[i].equalsIgnoreCase(agesucht[0])) {       //moeglicher startvergleichspunkt gefunden
          checking(i);                                      //i moelicher index in buchtext array
115       count++;
        }
117   }

119   if (count == 0) {
        System.out.println("Keine␣Loesung␣moeglich");
121   }
    }
123
    public static void checking(int j){  //j moelicher index in buchtext array
125
      int agesuchtl = agesucht.length;
127
      boolean failed = false;
129
      String part = new String();
131
      int spacesUsed = 0;   //die anzahl der _ in der gesuchten stelle
133

135   for (int i = 0; i < agesuchtl; i++) {
        if (achar[i]==0){
137       spacesUsed ++;
        }
139
        if (agesucht[i].equalsIgnoreCase(atext[i+j]) || achar[i]==0) {
141       part += atext[i+j] + "␣";
        } else {
143       failed = true;
        }
```

```
145       }
147       if (!failed) {
            System.out.println("Loesung␣gefunden!:␣" + part);
149       }
        }
151 }
```

Listing 1: Quellcode für Aufgabe 1: Stoerung von Dominik Luckert