Junioraufgabe 1: Parallelen

Team-ID: 00041

Team-Name: team@sebastianbrunnert

Bearbeiter/-innen dieser Aufgabe:
Sebastian Brunnert

5. September 2019

Inhaltsverzeichnis

Lösungsidee

Das Programm soll prüfen, ob ein vom Nutzer angegebener Text „parralel“ ist (siehe Aufgabenstellung). Als **Zusatzfuntkion** habe ich eingebaut, dass der Nutzer ebenfalls angegeben kann, wieviele erste Worte parallel sein müssen. Auf diese Idee bin ich gekommen, da in der gestellten Aufgabe nicht klar definiert ist, was die erste Hälfte ist.

Simulation

Um dieses Problem zu lösen, habe ich überlegt, wie ich (als Mensch) vorgehen würde, um zu prüfen, ob ein Text parallel ist. Mein Vorgehensweise wäre es beim ersten Wort des Textes anzufangen, die Anzahl n der Buchstaben dieses Wortes weiterspringen und so fortzufahren, bis ein Weiterspringen aufgrund eines Mangels an weiteren Worten nicht mehr möglich ist. Das Wort, bei dem ich nun bin, merke ich mir. Dieses Prozedere wiederhole ich für jedes Wort. Stimmt das Wort, welches ich beim aktuellen Durchgang erhalten habe, mit dem Wort, das ich mir aus dem ersten Durchgang gemerkt habe, nicht überein, so ist der Text nicht parallel. Komme ich bis zum letzten Wort, ab dem es nicht mehr möglich ist, der Anzahl der Buchstaben entsprechend viele Wörter im Text weiter zu springen, so ist der Text hingegen parallel. Exakt dies simuliert mein Programm mithilfe von Programmcode.

Umsetzung

Die Umsetzung des Programms geschieht in der Programmiersprache Java. Benötigt werden keine Programmbibliotheken, jedoch nutze ich das Build-Management-Tool Maven.

Zu Start des Programms werden über die von javax bereitgestellte Klasse JOptionPane die Nutzereingaben abgerufen. Zuerst der Text und im Anschluss die Range, wieviele erste Wörter des Textes „parallel“ sein müssen. Nun werden aus dem angegebenen Text alle Sonderzeichen herausgefiltert und alle Wörter in einen String-Array gespeichert. Es wird ebenfalls geprüft, ob die angegebene Range größer ist, als die Länge an Wörtern im Text. Ist dies der Fall, wird simpel die angebene Zahl auf das mögliche Maximum gesetzt.

Der nun folgende Vorgang wird für jedes Wort, das in der Range liegt ausgeführt: Es wird $n_{AktuellesWort}$ als die Zahl definiert, welche auch die Stelle des Wortes in dem Text ist. Nun wird eine normal unendliche Schleife (While-True-Schleife) gestartet, in der als erstes geprüft wird, ob $n_{AktuellesWort}$ + Länge des $n_{AktuellesWort}$en Wort größer ist als die Anzahl an Wörtern im Text.

Ist dies der Fall: ist das Endwort dieses Durchganges erreicht. Es wird deshalb geprüft, ob $s_{Endwort}$ existiert und, ob $s_{Endwort}$ mit dem erhaltenen Endwort übereinstimmt. Stimmen diese überein, so stoppt die While-True-Schleife und der gesamte Vorgang wird für das nächste Wort wiederholt (siehe S.2 Z.1). Existiert $s_{Endwort}$ nicht so wird dieser String zu dem erhaltenen Wort gesetzt. Stimmen diese aber nun nicht überein, so ist der Text nicht parallel und das Programm stoppt.

Ist dies nicht der Fall: wird $n_{AktuellesWort}$ zu $n_{AktuellesWort}$ + Länge des $n_{AktuellesWort}$en Wort gesetzt und die While-True-Schleife beginnt wieder.

Sollte das Programm nun den Vorgang für alle Wörter durchgegangen sein und ist dabei nicht gestoppt, so ist der Text parallel.

Beispiele

Wir rufen das Programm nun auf und geben diverse Werte an:

1	$ java -jar Junioaraufgabe 1 – Parallelen.jar Text: Es gingen zwei Parallelen ins Endlose hinaus, zwei kerzengerade Seelen und aus solidem Haus. Sie wollten sich nicht schneiden bis an ihr seliges Grab: Das war nun einmal der beiden geheimer Stolz und Stab. Doch als sie zehn Lichtjahre gewandert neben sich hin, da wards dem einsamen Paare nicht irdisch mehr zu Sinn. Warn sie noch Parallelen? Sie wußtens selber nicht, – sie flossen nur wie zwei Seelen zusammen durch ewiges Licht. Das ewige Licht durchdrang sie, da wurden sie eins in ihm; die Ewigkeit verschlang sie als wie zwei Seraphim. Range: 20 Ergebniss: Text ist parellel.

2	$ java -jar Junioaraufgabe 1 – Parallelen.jar Text: Das ist eine kleine versteckte Nachricht. Vielen Dank für das Bearbeiten meiner Lösungen :) Range: 10 Ergebniss: Text ist nicht parellel.

Quellcode

Der vollständige Quellcode, ist in dem Ordner Quellcode zu finden.

```
// Da Sonderzeichen keine Worte darstellen, werden diese aus dem Text genommen
text = text.replaceAll("[^a-zA-Z0-9\\s]", "");
// Alle Worte heraussuchen
String[] words = text.split("\\W+");

// Prüfe, ob angegebene Nummer größer ist als Menge an Wörtern im von Nutzer angegebenen Text
if(range > words.length){
   // Setze Nummer zur Textlänge
   range = words.length;
}

// Deklariere das Wort, welches das letzte Wort sein wird. Es wird nur eine Variable benötigt, da im Fall, dass es verschiedene Endwörter gibt, das Programm gestoppt wird, da Text nicht parallel ist
String endingWord = null;

// Gehe jedes Wort durch, welches in der angegebene Range liegt (für jedes muss das Endwort herausgesucht werden)
for(int i = 0; i < range; i++){
   // Suche das Wort
   String word = words[i];

   // Kopiere i (aus Schleife), da diese Variable verändert werden wird
   // current steht für den Index des Wortes, welches gerade geprüft wird
   int current = i;

   // Eine normal unendliche Schleife muss genutzt werden, da Text unendlich viele Zeichen haben kann
   while(true) {
      // Prüfe, ob ein Vorgang noch möglich wäre (wenn nein, ist das Endwort erreicht)
      if(words.length <= current+words[current].length()) {
         if(endingWord == null){
```

```
            // Sollte dies das erste Endwort sein, wird es als solches gespeichert
            endingWord = words[current];
        } else if(words[current] != endingWord){
            // Falls das erste Endwort nicht mit diesem überinstimmt, ist Parallelität nicht gewährleistet
            JOptionPane.showMessageDialog(null, "Der angegebene Text ist bis zum " + range + "ten Wort *nicht* parallel.", "Junioraufgabe 1 - Parallelen", JOptionPane.WARNING_MESSAGE);
            System.exit(0);
        }
        // Stoppe unendliche Schleife
        break;
    }
    // Setze den Index für das aktuelle Wort zu dem bereits vorhandenen Wort + der Länge des aktuellen Wort, wie in Aufgabenstellung beschrieben
    current += words[current].length();
  }
}

// Sollte nach diesem Algorithmus das Programm nicht gestoppt worden sein, ist der Text parallel
JOptionPane.showMessageDialog(null, "Der angegebene Text *ist* bis zum " + range + "ten Wort parallel.", "Junioraufgabe 1 - Parallelen", JOptionPane.PLAIN_MESSAGE);
```