Aufgabe 1: Hopsitexte

Team-ID: 00005

Team: BugBusters

Bearbeiter/-innen dieser Aufgabe:

Jonathan Salomo (Teilnahme-ID: 73504)

18. November 2024

Inhaltsverzeichnis

Lösungsidee

Diese Aufgabe lässt sehr viel Spielraum für die Lösung offen. Ich habe sie so verstanden, dass die Lösung ein Texteditor ähnliches Programm sein soll, welches es durch verschiedene Funktionen sehr einfach macht, einen Hopsitext zu schreiben.

Um dies umzusetzen, habe ich als Erstes überlegt, warum es so oft vorkommt, dass beide Spieler an der selben Stelle aus dem Text springen. Hierbei fällt auf, dass von einem Buchstaben, auf den einer der beiden Spieler springt, immer genau der selbe Weg aus dem Text genommen werden muss. Dies ist der Fall, da die Sprungdistanzen nicht wie bei anderen Spielen, wie zum Beispiel „Mensch ärgere dich nicht“, durch einen Würfel oder Ähnliches zufällig gemacht werden, zumindest einen gewissen Anteil an Zufall enthalten oder von Spielern taktisch gewählt werden können. Stattdessen hängen sie beim Texthopsen nur von dem Buchstaben ab, von dem ein Spieler los springt, und dem folgenden Text. Diese Eigenart des Spieles sorgt dafür, dass sobald beide Spieler an der selben Stelle im Text landen, sie immer auch von der selben Stelle heraus springen werden. Dieses Verhalten tritt bei längeren Texten dementsprechend auch öfter auf, da hier bei mehr Sprüngen die Wahrscheinlichkeit, einmal auf den selben Buchstaben zu springen, zunimmt.

Wenn wir also einen Hopsitext schreiben wollen, müssen wir nur dafür sorgen, dass nie beide Spieler auf die selbe Stelle springen, da sie dann auch nie von der selben Stelle aus dem Text springen werden.

Um es Zara möglichst einfach zu machen, dies zu erreichen, habe ich den „Hopsitext Editor 3000“ entwickelt. Dieser basiert stark auf der Lösung, die ich für die Junioraufgabe 2 eingereicht habe. Ich habe diese schon so geschrieben, dass sie den Weg, den die Spieler gehopst sind, in der selben Form eines Textes mit eingefärbten Buchstaben auszugeben vermag, wie er auf dem Aufgabenblatt zu finden ist.

Dies erreiche ich dort wie hier durch folgende Taktiken.

Die erste davon ist, dass ich zu Anfang den Text nicht in seiner Ursprungsform in den Algorithmus übergebe, sondern erst alles aus diesem herausfiltere, was kein Buchstabe ist. Dies ermöglichst es, dass ich im nächsten Schritt eindeutige Koordinaten für jeden einzelnen Buchstaben, also jeden möglichen Schritt, habe.

Anschließend setze ich die zwei Startpositionen, wie es die Aufgabe vorgibt, und beginne jeweils zu springen. Hierbei springe ich so lange, bis beide Spieler aus dem Text gesprungen sind, und merke mir jeden Buchstaben, auf und von dem sie springen. Bei jedem Sprung wird geprüft, welchen Wert der Buchstabe hat, auf dessen Koordinate gesprungen wurde, und dieser dann zu der vorherigen Koordinate addiert. Dieser Prozess simuliert das Hopsen entsprechend der Aufgabenstellung akkurat.

Nun gebe ich den ursprünglichen Text einmal wie in der Aufgabenstellung aus. Hierbei gehe ich einmal durch den kompletten Text und immer, wenn ich ein anderes Zeichen als in der Tabelle aufgeführt sehe, erhöhe ich einen Zähler. Wenn das aktuelle Zeichen jedoch ein Buchstabe aus der Tabelle ist, schaue ich, ob einer der beiden über diesen gesprungen ist, indem ich die Koordinate des Sprungs durch die Iteration im Text minus meinem Zähler für die anderen Zeichen berechne. Währenddessen färbe ich die Ausgabe entsprechend dem Beispiel der Aufgabenstellung ein, damit man sofort erkennen kann wo lang die Spieler gesprungen sind. Sobald sie auf gleiche Buchstaben springen färbe ich diese violett ein.

Um mit dem Hopsitext Editor 3000 zu arbeiten, beginnt man nun mit dem ersten Wort des Textes, das man schreiben will. Der Editor rechnet nun genauso wie grade beschreiben aus, was passieren würde, wenn man mit diesem Wort/Text Texthopsen spielen würde. Dies gibt er dann aus, indem er den gesamten Text mit eingefärbten Sprung-Positionen ausgibt. Zusätzlich dazu gibt er noch eine Zeile mit Nullen zurück, die jeweils für einen Buchstaben stehen, der beim Weiterschreiben an den Text angefügt wird. In diesen sind auch die nächsten Positionen markiert, auf die die beiden Spieler springen würden.

Nun ist es die Aufgabe von Zara sich ihr nächstes Wort beziehungsweise ihren nächsten Satzteil zu überlegen. Diesen sollte sie nun so weit eintippen, bis sie einen Buchstaben unter der ersten Null platziert, welche eingefärbt ist.

Wenn Zara dies getan hat, berechnet der Hopsitext Editor 3000 die nächsten Sprünge nach der selben Logik wie schon am Anfang. Dies Ablauf wiederholt sich solange, bis bei einer der Berechnungen beide Spieler auf das selbe Feld springen. Wenn dies passiert, setzt der Editor den Text auf die Version vor der letzten Eingabe zurück, die der Auslöser hierfür sein muss, wenn der Hopsitext Editor 3000 wie empfohlen verwendet wird. Dem Nutzer wird daraufhin mitgeteilt, dass er etwas Anderes schreiben muss, da es sonst einen Konflikt gibt, und dieser darf weiterschreiben.

Dies geht solange weiter, bis der Benutzer „end“ eintippt. Dann wird das Programm beendet und der Text ist fertig geschrieben.

Wenn man während der Bearbeitung irgendwann einen Schreibfehler einbaut, kann man die letzte Eingabe durch die Eingabe von „back“ wieder rückgängig machen.

Mit diesem Editor ist auch sehr gut möglich Hopsitexte in anderen Sprachen zu verfassen, solange man die Kommandos ändert und den Zeichensatz anpasst.

Umsetzung

Ich habe den Hopsitext Editor 3000 in Java objektorientiert geschrieben.

Der Editor beginnt damit, nach dem ersten Wort des Textes zu fragen und nach der Eingabe von diesem zu erklären, wie der Editor zu benutzten ist. Hiernach beginnt eine while-Schleife, in der der Text geschrieben wird. Diese läuft so lange, bis die „aktuelleEingabe“ „end“ ist.

In dieser Hauptschleife wird damit begonnen, die Methode „hopseText“ der „Texthopsen“ Junioraufgabe 2, die ich hierfür leicht modifiziert habe, aufzurufen.

Diese beginnt damit, im Converter den aktuellen Text zu verarbeiten. Durch die beiden String-Funktionen .toloercase() und .replaceall() aus der Standartbibliothek ersetze ich alle Zeichen bis auf Buchstaben. Diese speicher ich nun durch eine for-Schleife in einem Char-Array für einen schnellen Zugriff.

Nun habe ich meinen Hopser, dem ich den so überarbeiteten Text übergebe. Dieser beginnt damit, dass er ArrayLists für beide Teilnehmer erstellt, um ihre Sprünge zu speichern. Diese werden mit der Länge von dem Text durch 15 initialisiert, um Speicher zu sparen, da diese so im Durchschnitt für alle Sprünge reichen. Die 15 kommt hier daher, dass ich annehme, dass ein Sprung bei den vorgegebenen Sprungweiten von 1 – 30 im Durchschnitt 15 lang ist, was aufgrund der Häufigkeit der Buchstaben in der deutschen Sprache meist zutreffend sein wird[1]. Auch wenn dies bei einem Text nicht der Fall ist und mehr Sprünge gebraucht werden, muss die Liste auch nur einmal vergrößert werden. Nun initialisiere ich noch zwei Variablen mit den Startpositionen der Teilnehmer.

Um die Sprünge zu simulieren, verwende ich eine while-Schleife, in welcher fünf if-Abfragen gemacht werden. Als Erstes wird geschaut, ob der erste Spieler bereits gewonnen hat. Wenn dies

1 https://de.wikipedia.org/wiki/Buchstabenh%C3%A4ufigkeit

nicht der Fall ist, rufe ich eine HaschMap auf, die die Buchstaben auf in der Aufgabenstellung vorgegebenen Werte mapt. Durch die Map berechne ich nun die nächste Koordinate und simuliere so den Sprung. Dann mache ich noch eine if-Abfrage, welche kontrolliert, ob der Spieler noch im Text ist und, sollte dies nicht mehr der Fall sein, diesen als den Gewinner festlegt, solange der andere Spieler nicht bereits gewonnen hat. Entsprechend verfahre ich für den zweiten Teilnehmer. In der Dritten if-Abfrage kontrolliere ich nun, ob beide Teilnehmer noch im Text sind und beende die Schleife, wenn dies nicht mehr der Fall ist.

Um nun den Text in der Form auszugeben, wie er auf dem Aufgabenblatt angegeben ist, gehe ich einmal mit einer for-Schleife durch den ursprünglichen Text. Hierbei kontrolliere ich als Erstes, ob ich bei der aktuellen Koordinate ein Buchstaben oder ein anderes Zeichen vorliegen haben. Wenn es sich um ein anderes Zeichen handelt, printe ich es auf die Konsole und erhöhe einen Zähler für alle „nicht Buchstaben“, an denen ich vorbeigekommen bin. Wenn es sich um einen Buchstaben handelt, finde ich seine Koordinate im System des Hopsers, indem ich seine aktuelle Koordinate im Text minus all die anderen Zeichen nehme. Durch diese finde ich dann mit einem Aufruf des Hopsers heraus, ob dieser Buchstabe einzufärben ist und setze dies dementsprechend um.

Am Ende der Berechnung übergebe ich den Hopser an die Hauptschleife.

Innerhalb der Hauptschleife wird nun durch eine if-Abfrage kontrolliert, ob die beiden Spieler über die selben Felder springen. Wenn dies der Fall ist, wird der Text auf die vorherige Version zurückgesetzt, welche immer in „lastText“ gespeichert wird, und der Benutzer wird durch eine Ausgabe informiert. In jedem andern Fall werden nun die Nullen ausgegeben und passend eingefärbt. Hierfür itteriere ich mit einer for-Schleife einmal über jeden Buchstaben, der theoretisch nach dem Ende der Textes angefügt werden könnte, bis ich zum Buchstaben direkt hinter der größeren der beiden aktuellen Positionen der Spieler gelange, und färbe die Positionen wieder farblich ein.

Danach update ich die „aktuelleEingabe“ durch die Funktion .readLine() der Console aus der Java Standartbibliothek. Als Letztes mache ich noch eine if-Abfrage, die kontrolliert, ob der Benutzer den Befehl „back“ eingegeben hat und, wenn dies der Fall ist, den Text auf den letzten Text zurücksetzt, andernfalls den „leztenText“ auf den aktuellen setzt und an den bisherigen Text die „aktuelleEingabe“ anfügt.

Wenn die Schleife durch die Eingebe von „end“ beendet wird beendet sich das Programm.

Beispiele

```
PS D:\Documents\Programiren\Informatik Wettbewerb\BWINF 2024\Runde 1\Hopsitexte>  & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCo
INF 2024\Runde 1\Hopsitexte\bin' 'Hopsitexte'
--------------------------------------------------------------------------------------------------------------------------
Willkommen zum Hopsitext Editor 3000!
--------------------------------------------------------------------------------------------------------------------------
Mit diesem Programm kannst du ganz einfach einen Hopsitext schreiben.
Ein Hopsitext ist ein spezieller Text, bei dem sichergestellt wird, dass die Spieler beim Texthopsen nie auf derselben Stelle landen.
--------------------------------------------------------------------------------------------------------------------------
So funktioniert es:
- Beginne, indem du das erste Wort oder den ersten Satzteil in die Eingabezeile eingibst.
- Der Editor zeigt dir dann die Positionen an, die beim Texthopsen relevant sind.
- Schreibe deinen Text weiter, bis du zu einer markierten Position gelangst.
- Wenn ein Konflikt auftritt (beide Spieler landen an derselben Stelle), wirst du benachrichtigt und kannst den letzten Schritt korrigieren.
- Du kannst jederzeit den letzten Schritt mit 'back' rückgängig machen.
- Um deinen Hopsitext fertigzustellen, gib 'end' ein.
--------------------------------------------------------------------------------------------------------------------------
Starte jetzt mit deinem ersten Wort oder Satzteil:
Vielen Dank für Ihre sorgfältige Korrektur
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur
--------------------------------------------------------------------------------------------------------------------------
00000000000000
, ich hoffe mein Editor gefällt ihnen.
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur, ich hoffe mein Editor gefällt ihnen.
--------------------------------------------------------------------------------------------------------------------------
Bitte verwende ein anderes Wort da dein Text sonst kein Hopsitext sein kann.
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur
--------------------------------------------------------------------------------------------------------------------------
00000000000000
, ich hoffe mein Programm gefällt ihnen.
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur, ich hoffe mein Programm gefällt ihnen.
--------------------------------------------------------------------------------------------------------------------------
000000000000
 Ich freue mich schon auf die zweite Runde (;
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur, ich hoffe mein Programm gefällt ihnen. Ich freue mich schon auf die zweite Runde (;
--------------------------------------------------------------------------------------------------------------------------
000000000000000000
back
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur, ich hoffe mein Programm gefällt ihnen.
--------------------------------------------------------------------------------------------------------------------------
000000000000
 Ich freue mich schon sehr auf die zweite Runde (;
--------------------------------------------------------------------------------------------------------------------------
Vielen Dank für Ihre sorgfältige Korrektur, ich hoffe mein Programm gefällt ihnen. Ich freue mich schon sehr auf die zweite Runde (;
--------------------------------------------------------------------------------------------------------------------------
000000000000000000
end
PS D:\Documents\Programiren\Informatik Wettbewerb\BWINF 2024\Runde 1\Hopsitexte>
```

Quellcode

Hopsitexte

```
public class Hopsitexte {
    public static void main(String[] args) {

        Console console = System.console();


        Texthopsen texthopsen = new Texthopsen();


        System.out.println(ANSI_YELLOW + "Willkommen zum Hopsitext Editor 3000!" + ANSI_RESET);

        System.out.println("Mit diesem Programm kannst du ganz einfach einen Hopsitext erstellen.");
        System.out.println("Ein Hopsitext ist ein spezieller Text, bei dem sichergestellt wird, dass
die Spieler beim Texthopsen nie auf derselben Stelle landen.");

        System.out.println("So funktioniert es:");
        System.out.println("- Beginne, indem du das erste Wort oder den ersten Satzteil in die
Eingabezeile schreibst.");
        System.out.println("- Der Editor zeigt dir dann die Positionen an, die beim Texthopsen
relevant sind.");
        System.out.println("- Schreibe deinen Text weiter, bis du zu einer markierten Position
gelangst.");
        System.out.println("- Wenn ein Konflikt auftritt (beide Spieler landen an derselben Stelle),
wirst du benachrichtigt und kannst den letzten Schritt korrigieren.");
        System.out.println("- Du kannst jederzeit den letzten Schritt mit 'back' rückgängig
machen.");
        System.out.println("- Um deinen Hopsitext fertigzustellen, gib 'end' ein.");

        System.out.println("Starte jetzt mit deinem ersten Wort oder Satzteil:");
        String aktuelleEingabe = scanner.nextLine();

        String lastText = "";
        String text = aktuelleEingabe;
        while (!aktuelleEingabe.equals("end")) {
            Hopser h = new Hopser();
            try {
            h = texthopsen.hopseText(text);
            } catch (java.lang.ArrayIndexOutOfBoundsException e) {
                System.out.println("Bitte überprüfe deinen bisherigen Text und passe ihn an, um den
Fehler zu vermeiden.");
                break;
            }

            if (h.currendPos1 == h.currendPos2 || texthopsen.selbesFeld){
                System.out.println("Bitte verwende ein anderes Wort da dein Text sonst kein
Hopsitext sein kann.");
                text = lastText;
            } else{

                for (int i = texthopsen.extractedLetters.length; i <= Math.max(h.currendPos1+1,
h.currendPos2+1); i++){
                    if(i == h.currendPos1){
                        System.out.print(ANSI_RED + "0" + ANSI_RESET);
                    }else if(i == h.currendPos2){
                        System.out.print(ANSI_BLUE + "0" + ANSI_RESET);
                    } else{
                        System.out.print("0");
                    }
                }
                System.out.println();


                aktuelleEingabe = scanner.nextLine();

                if (aktuelleEingabe.equals("back")){
                    text = lastText;
```

```
                } else{
                    lastText = text;
                    text = text + aktuelleEingabe;

                }
            }
        }
        scanner.close();
    }
}
```

Texthopsen

```
public class Texthopsen {

        Converter converter = new Converter();
        extractedLetters = converter.textToLetters(text);

        Hopser hopser = new Hopser();
        hopser.hopsen(0, 1, extractedLetters);
        for (int i = 0; i < text.length(); i++) {
            if (!Character.isLetter(text.charAt(i))){
                System.out.print(text.charAt(i));
                irelewanteZeichen++;
            } else{
                boolean pos1InRange = (posi1 < hopser.pos1.size());
                boolean pos2InRange = (posi2 < hopser.pos2.size());

                if (pos1InRange && pos2InRange && hopser.pos1.get(posi1) == i-irelewanteZeichen &&
hopser.pos2.get(posi2) == i-irelewanteZeichen) {
                    System.out.print(ANSI_PURPLE + text.charAt(i) + ANSI_RESET);
                    selbesFeld = true;
                    if (posi1 < hopser.pos1.size()) posi1++;
                    if (posi2 < hopser.pos2.size()) posi2++;
                } else if (pos1InRange && hopser.pos1.get(posi1) == i-irelewanteZeichen) {
                    System.out.print(ANSI_RED + text.charAt(i) + ANSI_RESET);
                    if (posi1 < hopser.pos1.size()) posi1++;
                } else if (pos2InRange && hopser.pos2.get(posi2) == i-irelewanteZeichen) {
                    System.out.print(ANSI_BLUE + text.charAt(i) + ANSI_RESET);
                    if (posi2 < hopser.pos2.size()) posi2++;
                } else {
                    System.out.print(text.charAt(i));
                }
            }
        }

        return hopser;
    }
}
```

Hopser

```
public class Hopser {

    private static final Map<Character, Integer> LETTER_TO_NUMBER_MAP = createMap();

    private static Map<Character, Integer> createMap() {
        Map<Character, Integer> map = new HashMap<>();

        map.put('a', 1);
        map.put('b', 2);
        map.put('c', 3);
        map.put('d', 4);
        map.put('e', 5);
        map.put('f', 6);
        map.put('g', 7);
        map.put('h', 8);
        map.put('i', 9);
        map.put('j', 10);
        map.put('k', 11);
        map.put('l', 12);
        map.put('m', 13);
        map.put('n', 14);
        map.put('o', 15);
        map.put('p', 16);
        map.put('q', 17);
        map.put('r', 18);
        map.put('s', 19);
        map.put('t', 20);
        map.put('u', 21);
        map.put('v', 22);
        map.put('w', 23);
        map.put('x', 24);
        map.put('y', 25);
        map.put('z', 26);
        map.put('ä', 27);
        map.put('ö', 28);
        map.put('ü', 29);
        map.put('ß', 30);

        return map;
    }

    public int hopsen(int start1, int start2, char[] extractedLetters){
        int win = 0;

        pos1 = new ArrayList<>(extractedLetters.length / 15);
        pos2 = new ArrayList<>(extractedLetters.length / 15);

        currendPos1 = start1;
        currendPos2 = start2;
        while (true){

                if (win != 1){
                pos1.addLast(currendPos1);
                currendPos1 += LETTER_TO_NUMBER_MAP.get(extractedLetters[currendPos1]);
                nededjumps1++;
                }
                if (currendPos1 > extractedLetters.length-1 && win == 0){
                    win = 1;
                }

                if (win != 2){
                pos2.addLast(currendPos2);
                currendPos2 += LETTER_TO_NUMBER_MAP.get(extractedLetters[currendPos2]);
                nededjumps2++;
                }
                if (currendPos2 > extractedLetters.length-1 && win == 0){
                    win = 2;
                }

                if (win != 0 && currendPos2 > extractedLetters.length-1 && currendPos1 >
extractedLetters.length-1){
```

```
                    break;
                }

        }
        return win;
    }
```

Converter

```
public class Converter{

    public char[] textToLetters(String text) {
        String cleanText = text.toLowerCase().replaceAll("[^a-zäöüß]", "");

        char[] letters = new char[cleanText.length()];
        for(int i = 0; i < letters.length; i++) {
            letters[i] = cleanText.charAt(i);
        }

        return letters;
    }
}
```