Aufgabe 1: Hopsitexte

Team-ID: 00178

Team-Name: Team-Name

Bearbeiter/-innen dieser Aufgabe:
Matthew Greiner

November 17, 2024

Contents

1 Lösungsidee / Ansatz

1.1 Anforderungen an das Programm

In dieser Aufgabe geht es darum, ein Programm zu erstellen, das Zara hilft, deutsche Hopsitexte zu erstellen. Um ein Programm zu gestalten, damit Zara besonders gut damit arbeiten kann, müssen meiner Meinung nach, folgende Kriterien für das Programm erfüllt sein:

1. Das Programm muss ein graphisches Nutzerinterface mit einem Textfeld haben, in dem Zara ihren Text tippen kann.
2. Die Stellen, an denen die Sprünge stattgefunden haben, müssen wie in den Beispieltexten, rot für den ersten Spieler und blau für den zweiten Spieler, gefärbt sein.
3. Da es ein deutscher Text sein soll, müssen Umlaute wie "ä, ö, ü" und "ß" auch unterstützt werden.
4. Wenn der Text kein Hopsitext ist, muss die Stelle, an dem sich die Spieler treffen, farblich markiert sein, damit Zara weiß, dass sie Änderungen vornehmen muss.
5. Es muss anzeigen, ob der eingegebene Text ein "Hopsitext" ist oder nicht.
6. Die Ausgabe des Programmes und die farbliche Markierungen des Eingabetextes müssen dynamisch und automatisch mit dem Tippen von Zara geschehen und Zara kann an beliebigen Stellen im Eingabetext Änderungen vornehmen (Quality of life).

Grobe Struktur

Auf der linken Seite des Programmes soll Zara ihren Text tippen können und auf der rechten Seite soll dieser Text an den Sprüngen der Spieler farblich markiert, - wie im Beispieltext, angezeigt werden. So sieht Zara beim tippen, ob ihre aktuelle Eingabe ein Hopsitext ist oder nicht. Zusätzlich soll oberhalb des Textfeldes deutlich angezeigt werden, ob es sich um einen Hopsitext handelt. Wenn diese Kriterien erfüllt sind, hat Zara ein Programm, in der sie einen Text schreiben kann, das ihr beim tippen anzeigt, ob ihre aktuelle Eingabe ein Hopsitext ist oder nicht. Falls es kein Hopsitext ist, sieht Zara das sofort und kann ihren Text bzw. einfach die letzen Worte mit dem Sprung ändern.

2 Umsetzung

Um dieses Programm zu erstellen, habe ich JavaFX als GUI Library verwenden, welches mir grundlegende Bausteine für das Interface zur Verfügung stellt. Zur Implementierung werden die in (1.1) angesprochenen Anforderungen einzeln umgesetzt:

1. Zu Beginn erstelle ich mit JavaFX Modulen ein Textfeld ("TextArea") auf der linken Seite des Programms, eine formatierte Textanzeige ("Textflow") auf der rechten Seite, in der die Hopsisprünge markiert werden und einen Statustext ("Text") mit dem Ergebnis ganz oben. Das Textfeld bekommt einen Listener, der jedes mal gerufen wird, wenn sich der Text im Textfeld ändert. So wird jedes mal das Ergebnis und der formatierte Text auf der rechten Seite geupdated, sobald sich die Texteingabe ändert.

2. Eine Methode wird erstellt, die die Texthops-Sprünge ab einer Startposition simuliert. Diese Methode wird zweimal aufgerufen, jeweils einmal für die Roten und die Blauen Sprünge. Die Methode funktioniert wie folgt:
    1. Der Eingabetext als String und die Farbe, mit der gesprungen werden soll, wird übergeben.
    2. Die ersten Zeichen, die keine Buchstaben sind, werden bei der Simulation der Texthops-Sprünge übersprungen und bei dem ersten Zeichen, das ein Buchstabe ist, wird begonnen.
    3. Dieses Zeichen wird mit der übergebenen Farbe gefärbt und zu einer Collection aus Zeichen (TextFlow) hinzugefügt. Die Position des Buchstabens im Alphabet wird sich gemerkt; z.B. als n.
    4. Dann werden die nächsten n Buchstaben übersprungen. Alle Zeichen, die keine Buchstaben sind, werden ignoriert. Alle Zeichen, inklusive der Nicht-Buchstaben werden ungefärbt zu dem TextFlow hinzugefügt.
    5. Anschließend wird der Buchstabe bei dem wir nach den Sprüngen angekommen sind, mit der übergeben Farbe gefärbt und zum Textflow hinzugefügt. Diese Schleife (3. - 5.) wird wiederholt, bis der Inputtext vollständig geparst wurde.
    6. Das resultierende TextFlow enthält nun den gesamten Inputtext, wobei die Hopsisprünge farblich markiert sind. Dieses Textflow wird zurückgegeben.

3. Um die Sprungweite eines Buchstabens gleich weit wie ihre Position im Alphabet zu machen und Umlaute auch zu beachten, muss eine Methode erstellt werden, die die zugehörige Sprungweite von einem übergebenen Zeichen zurück gibt:
Wenn das Zeichen ein deutsches Sonderzeichen ist, wird eine festgelegte Position zurückgegeben. Wenn es sich um einen normalen großen Buchstaben handelt, kann man um die Position im Alphabet zu bekommen, den ASCII-Wert von dem Zeichen mit dem ASCII-Wert von "A" subtrahieren und 1 addieren. Somit ist A=1, B=2, etc. Kleinbuchstaben funktionieren analog, nur wird hier der ASCII-Wert von "a" abgezogen. Alle anderen Zeichen geben -1 zurück.

4. Um die Stelle zu markieren, an der sich beide Spieler treffen (z.B. in **lila**), wird die Methode aus (2.) verwendet. Sie wird aufgerufen, um die Sprünge von beiden Spielern zu berechnen und in einem resultierendem TextFlow farblich zu markieren. Nun müssen diese beiden TextFlows verglichen werden. Wenn an einer bestimmten Stelle beide Buchstaben in beiden TextFlows farblich markiert sind, bedeutet das, dass sich die Spieler an dieser Stelle treffen. Daher muss für dieses Teilproblem eine Methode erstellt werden, die diese beiden TextFlows vereinigt und wenn sich die Spieler treffen, farblich markiert (lila), und dies als Ergebnis dem Nutzer anzeigt.

Dies geht indem man über Zeichen in den beiden TextFlows iteriert und überprüft, ob ein Buchstabe in beiden TextFlows farblich markiert ist. Wenn ja, ist an diesem Buchstaben eine Kollision von den Sprüngen, und wird im Ergebnistextflow **lila** markiert. Wenn keine Kollision an einem Buchstaben vorherrscht, wird der farblich markierte Buchstaben aus einem der beiden TextFlows übernommen. Wenn an einer Stelle kein Buchstabe farblich markiert ist, wird das Zeichen ungefärbt zum Resultattextflow hinzugefügt.

5. Wenn die Methode aus (4.) an beliebiger Stelle lila anzeigt, ist der übergebenen Text kein Hopsitext mehr. Dies wir über dem Nutzereingabefeld bekenntlich gemacht.

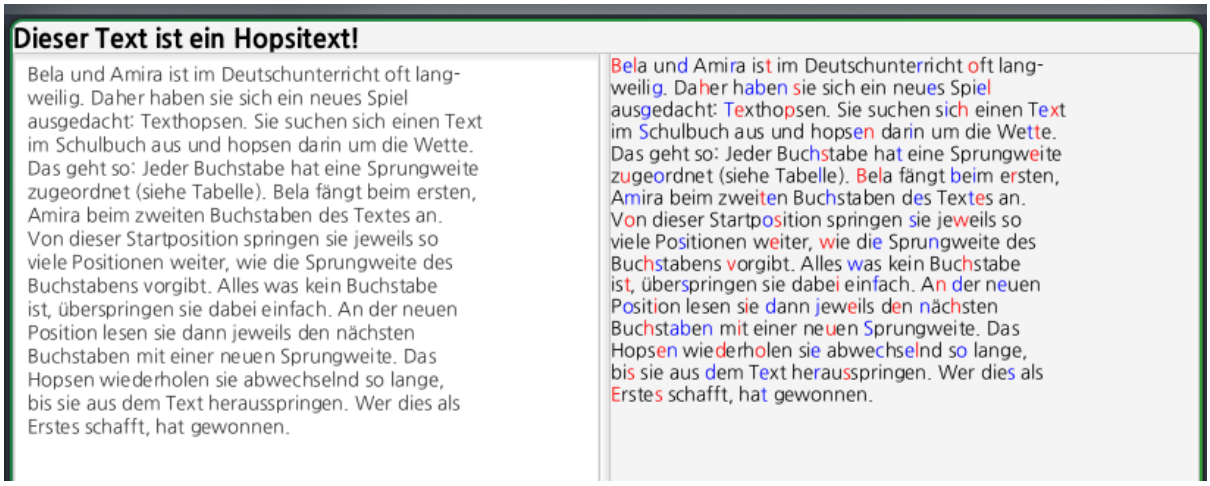
6. Das von oben genannte wird jedes mal ausführt, wenn sich das Eingabefeld ändert.

3 Beispiele

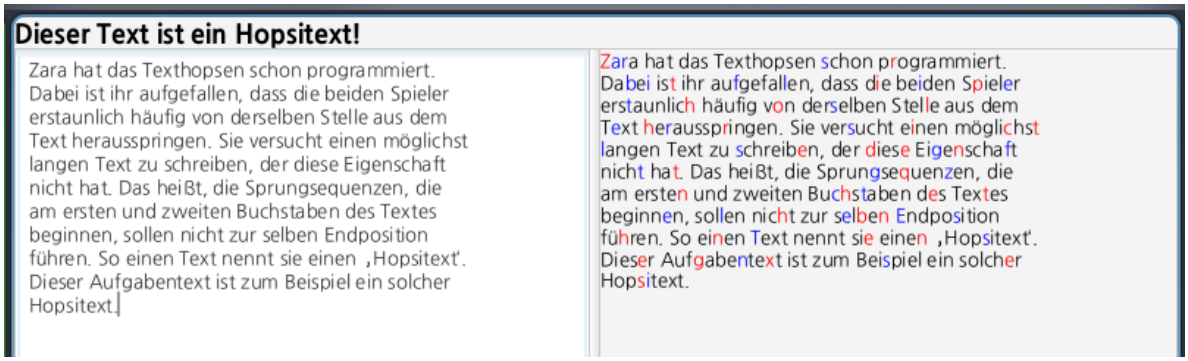
3.1 Beispiele BwInf-Aufgabenangaben

Hier ist das Progamm mit den Angabetexten der Junioraufgabe 2 und der Aufgabe 1 als Eingabetext.

Angabe aus Junioraufgabe 2

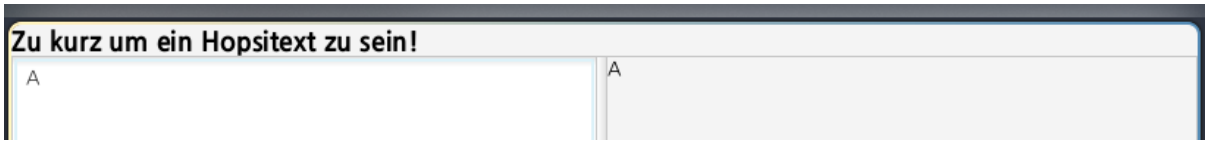


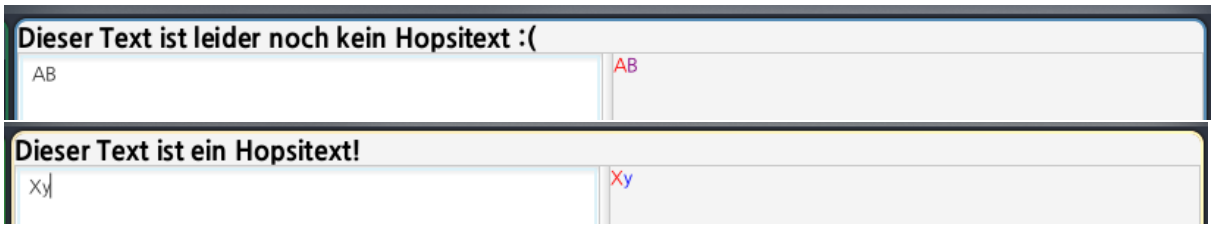
Angabe aus Aufgabe 1



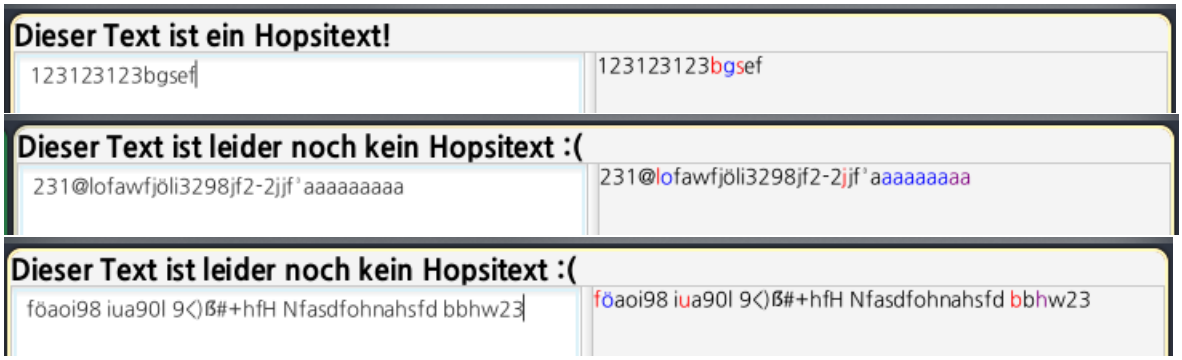
3.2 Eigene Beispiele / Edgecases

Kurze Texte

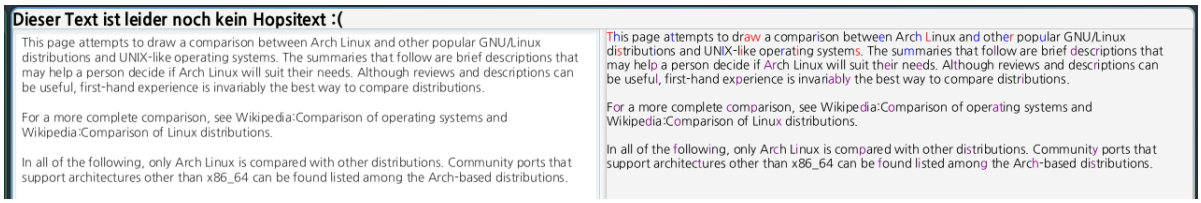


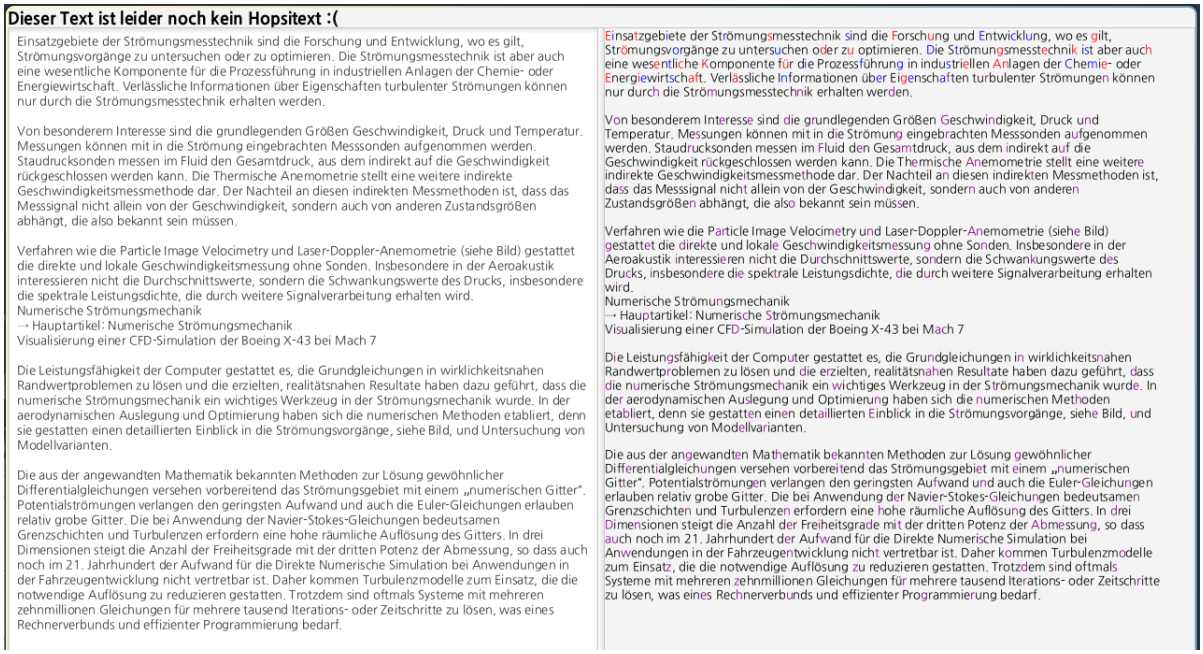


Random Input



Lange Texte





4 Quellcode

Simuliert Hopsitext Sprünge - Entspricht der Methode aus (2.2)

```
/**
* Simulates Hopsitext-jumps
*
* @param input - Inputtext as String
* @param startIndex - Starting Position of Player
* @param color - Color of Player
* @return TextFlow
*/
private TextFlow jump(String input, int startIndex, Color color) {
    // Textflow is a collection of textnodes
    TextFlow tf = new TextFlow();
    int nextIndex = 0;
    char currentChar = '␣';
    // Textnode is a single character, because characters need to be colored
    // individually
    Text textNode = new Text();

    // Skip over first non-alphabetic characters
    int i = 0;
    for (; i < input.length(); i++) {
        if (getAlphabetPosition(input.charAt(nextIndex)) == -1) {
            textNode = new Text(String.valueOf(input.charAt(nextIndex)));
            tf.getChildren().add(textNode);
            nextIndex++;
        } else {
            break;
        }
    }
    nextIndex += startIndex;

    if (nextIndex == (1 + i) && !input.isEmpty()) {
        textNode = new Text(String.valueOf(input.charAt(0)));
        tf.getChildren().add(textNode);
    }

    // Main loop
    while (nextIndex < input.length()) {
        currentChar = input.charAt(nextIndex);
        textNode = new Text(String.valueOf(currentChar));

        // Character that needs to be colored, beause jump starts on this char
        if (getAlphabetPosition(currentChar) != -1) {
            // Display character in specified color
            textNode.setFill(color);
            tf.getChildren().add(textNode);

            int alphabetPosition = getAlphabetPosition(currentChar);
            int count = 0;

            // Skip non-alphabetic characters and move forward by the alphabet position
            while (count < alphabetPosition) {
                nextIndex++;
                if (nextIndex >= input.length()) {
                    break;
                }
                // If Character is an alphabetic character
                if (getAlphabetPosition(input.charAt(nextIndex)) != -1) {
                    count++;
                    // Add uncolored character
                    if (count != alphabetPosition) {
                        textNode = new Text(String.valueOf(input.charAt(nextIndex)));
                        tf.getChildren().add(textNode);
                    }
                    // If Character is non alphabetic -> skip it and don't increment
    loop counter
                } else {
                    // Add uncolored character
                    textNode = new Text(String.valueOf(input.charAt(nextIndex)));
```

```
                    tf.getChildren().add(textNode);
                }
            }
        } else {
            textNode = new Text(String.valueOf(input.charAt(nextIndex)));
            // textNode.setFill(color);
            System.out.println("If this gets executed, then there is something wrong");
            tf.getChildren().add(textNode);
            nextIndex++;
        }
    }
    return tf;
}
```

Vereinigt zwei TextFlows - Entspricht der Methode aus (2.4)

```
/**
* Merges two TextFlows into one with merged colors of two players. Jump
* collisions are colored purple
*
* @param tf1 - first player jumps
* @param tf2 - second player jumps
* @return TextFlow
*/
private TextFlow mergeTextFlows(TextFlow tf1, TextFlow tf2) {
    TextFlow result = new TextFlow();
    int count = 0;

    // Loops over every character
    for (int i = 0; i < tf1.getChildren().size(); i++) {
        Text text1 = (Text) tf1.getChildren().get(i);
        Text text2 = (Text) tf2.getChildren().get(i);

        Color color1 = (Color) text1.getFill();
        Color color2 = (Color) text2.getFill();

        Text mergedText = new Text(text1.getText());

        // Result Color of current character
        Color finalColor = determineColor(color1, color2);
        if (finalColor.equals(Color.PURPLE)) {
            output.setText("Dieser Text ist leider noch kein Hopsitext :(");
            count = 1;
        } else {
            if (count == 1) {
                output.setText("Dieser Text ist leider noch kein Hopsitext :(");
            } else {
                output.setText("Dieser Text ist ein Hopsitext!");
            }
        }
        mergedText.setFill(finalColor);

        // Display result to user
        result.getChildren().add(mergedText);
    }
    return result;
}
```

Sprungweite aus Buchstaben ermitteln - Entspricht der Methode aus (2.3)

```
/**
* Return the position in the alphabet of the character. Upper and lowercase
* characters are treated the same. German "Umlaute" also return a valid
* position. Everything else returns -1
*
* @param c - Character to check
* @return int - Position in the alphabet
*/
private int getAlphabetPosition(char c) {
    switch (c) {
        case 'ä':
        case 'Ä':
            return 27;
        case 'ö':
        case 'Ö':
            return 28;
        case 'ü':
        case 'Ü':
            return 29;
        case 'ß':
            return 30;
        default:
            break;
    }

    if (Character.isUpperCase(c) && upperCase.contains(c)) {
        return c - 'A' + 1; // A=1, B=2, ..., Z=26
    } else if (Character.isLowerCase(c) && lowerCase.contains(c)) {
        return c - 'a' + 1; // a=1, b=2, ..., z=26
    } else {
        return -1;
    }
}
```

Nutzung der oben gezeigten Methoden

```
/**
* Receives Text input from the use and displays the processed version with
* correct colors for the jumps
*
* @param input - Text from the input field
*/
private void processText(String input) {
    textFlow.getChildren().clear();

    if (input.isEmpty()) {
        output.setText("Ist␣dein␣Text␣ein␣Hopsitext?␣Tippe␣in␣das␣Textfeld␣:)");
    } else if (input.length() == 1) {
        textFlow.getChildren().add(new Text(input));
        output.setText("Zu␣kurz␣um␣ein␣Hopsitext␣zu␣sein!");
    } else {
        textFlow.getChildren()
                .addAll(mergeTextFlows(jump(input, 0, Color.RED), jump(input, 1,
    Color.BLUE)).getChildren());
    }
}
```