Tag 2

- Variante 1

Hilfestellungen Variante 1

- Du kannst/musst exzessiv die split("...")-Methode von Java nutzen. Damit zerteilst du einen String an jedem gefundenen Trennzeichen, sodass ein String-Array entsteht. Beispiel: `line.split(":")` erzeugt ein Array der Form ["Game 1", "4 red, 1 green, 15 blue; 6 green UND SO WEITER"]. Das Array enthält also zwei Strings: Erstens den Teil VOR dem Doppelpunkt, und zweitens den gesamten restlichen Teil hinter dem Doppelpunkt.
- Die split("...")-Methode kannst du mehrmals direkt hintereinander aufrufen und zwischendurch mit dem üblichen Index-Zugriff in eckigen Klammern auf einen bestimmten String des String-Arrays zugreifen. Beispiel: Der Befehl `line.split(":")[1].split(";")` gibt dir ein String Array, bei dem jeder String ein Herausziehen des Elfen darstellt. Das Ergebnis ist dann also z. B. die Form ["4 red, 1 green, 15 blue", "6 green, 2 red, 10 blue", "..."]
- Unterteile jeden dieser Strings noch einmal mehr (an einem weiteren Satzzeichen), um jede einzelne Farbe zu erhalten.
- Nun kannst du für eine einzelne Farbe (z. B. `3 green`) prüfen, ob diese Anzahl für diese Farbe erlaubt ist. Dazu kannst du den String ein weiteres Mal am Leerzeichen zerteilen und für den String-Vergleich `equals()` benutzen.
- Es empfiehlt sich immer wieder vorsorglich die `strip()` Methode zu nutzen. Diese entfernt Whitespace-Character (also z. B. ein Leerzeichen) am Anfang eines Strings und am Ende.
- Nutze eine boolean-Variable, um festzuhalten, ob du in der aktuellen Zeile bereits eine illegale Anzahl einer Farbe gezogen hast. Dann kannst du nämlich früher die Schleifen unterbrechen (`break;`) und weißt am Ende auch, ob denn die aktuelle Zeilennummer aufaddiert werden muss oder nicht.

Lösungsvorschlag 1

```
private boolean colorAmountAllowed(String s) {
    String text = s.strip().split(" ")[1].strip();
    int amount = Integer.parseInt(s.strip().split(" ")[0].strip());

    if (text.equals("red") && amount > 12) {
        return false;
    } else if (text.equals("green") && amount > 13) {
        return false;
    } else if (text.equals("blue") && amount > 14) {
        return false;
    }
    return true;
}

public int partOne() {
    int summe = 0;

    for (String line : inputLines) {
        // Speichere die Game ID
```

```
        int gameID = Integer.parseInt(line.split(":")[0].split(" ")[1]);
        boolean possibleGame = true;

        // Zerteile die line in die einzelnen "reveals"
        String[] reveals = line.split(":")[1].split(";");
        for (String reveal : reveals) {

            // Zerteile jeden Reveal in die einzelnen Farben
            String[] colors = reveal.split(",");
            for (String color : colors) {

                // Prüfe jede Farben-Häufigkeit-Kombination
                if (!colorAmountAllowed(color)) {
                    possibleGame = false;
                    break;
                }
            }
            if (!possibleGame) {
                break;
            }
        }

        if (possibleGame) {
            summe += gameID;
        }

    }

    return summe;
}
```

From:
https://info-bw.de/ -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:aoc:aco2023:day2:start?rev=1701513099

Last update: **02.12.2023 10:31**