- Variante 1

Hilfestellung Teil 1

- Es verschiedene Möglichkeiten die wenigen Zahlen zu speichern. In dieser Lösung empfehle ich, alle Zahlen der ersten Zeile und alle Zahlen der zweiten Zeile jeweils in einem int-Array zu speichern. Diese Arrays kannst du z. B. `times` und `distances` nennen. Danach kannst du mit einer for-Schleife über die Werte iterieren. Mit deinem Laufindex (z. B. `i`) kannst du sicherstellen, dass du immer auf die zueinander gehörenden Zeiten und Distanzen zugreifst.
- Für jedes Wertepaar (Zeit und Distanz) musst du nun für jede mögliche 'Knopf-Drück-Zeit' von 0 bis 'Zeit' ms prüfen, ob die zurücklegbare Strecke größer als die eingelesene Distanz ist. Für die zurücklegbare Strecke gilt nach den Regeln der Physik: `s=v*t`. `v` ist die Geschwindigkeit, welche sich direkt aus der 'Knopf-Drück-Zeit' ergibt. `t` ist dann die noch übrige Zeit, um die Strecke zurückzulegen (also die eingelesene Zeit *minus* 'Knopf-Drück-Zeit'). Wenn das Ergebnis `v` größer ist als die eingelesene Distanz, dann wird der Counter für das Wertepaar um eins erhöht.
- Am Ende müssen alle Counter zusammenmultipliziert werden.

Lösungsvorschlag

```
private int[] numbersToIntArray(String numbersStr) {
    String[] numbers = numbersStr.trim().split("\\s+");
    int[] result = new int[numbers.length];

    for (int i = 0; i < numbers.length; i++) {
        result[i] = Integer.parseInt(numbers[i]);
    }

    return result;
}



public int partOne() {
    int[] times = numbersToIntArray(inputLines.get(0).split(":")[1].trim());
    int[] distances =
numbersToIntArray(inputLines.get(1).split(":")[1].trim());

    int result = 1;

    for (int i = 0; i < times.length; i++) {
        int numberOfWaysToBeat = 0;
        for (int buttonTime = 0; buttonTime < times[i]; buttonTime++) {
            if ((buttonTime * (times[i]-buttonTime)) > distances[i]) {
                numberOfWaysToBeat++;
            }
        }
        result *= numberOfWaysToBeat;
    }
```

```
    return result;
}
```

Hilfestellung Teil 2

- Teil 2 ist tatsächlich noch einiges einfacher als Teil 1. Dadurch, dass die erste und zweite Zahl jeweils als eine Zahl interpretiert wird, kann das Array wegfallen und damit kann auch eine Schleife wegfallen.
- Wichtig ist nun allerdings, dass man beide Zahlen als `long` speichert, da die Zahlen den Wertebereich von int andernfalls überschreiten können.
- Um aus den ersten Zahlen einer Zeile eine gemeinsame Zahl zu machen bevor man sie parsed, kann man alle Leerzeichen zwischen den Ziffern entfernen mit `replaceAll(" ","")`. Damit wird jedes Leerzeichen durch 'nix' ersetzt.

Lösungsvorschlag

```
public long partTwo() {
    long time = Long.parseLong(inputLines.get(0).split(":")[1].replaceAll(" ", ""));
    long distance = Long.parseLong(inputLines.get(1).split(":")[1].replaceAll(" ", ""));

    long numberOfWaysToBeat = 0;
    for (long buttonTime = 0; buttonTime < time; buttonTime++) {
        if ((buttonTime * (time-buttonTime)) > distance) {
            numberOfWaysToBeat++;
        }
    }

    return numberOfWaysToBeat;
}
```

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

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:aoc:aco2023:day6:start?rev=1701855182

Last update: **06.12.2023 09:33**