Day 2: Red-Nosed Reports

Teil 1

Auch Tag 2 ist relativ einfach. Du benötigst keine besonderen Java-Befehle außer der `split()`-Methode aus Tag 1.

Tipps zur Vorgehensweise

- Speichere jede Zeile in einem int-Array oder in einer ArrayList<Integer>
- Ich empfehle, 3 separate Methoden zu erstellen, die für ein übergebenes Java-Array jeweils prüfen, ob:
  - ... alle Zahlen nur aufsteigend sind
  - ... alle Zahlen nur absteigend sind
  - ... die Differenz aller aufeinander folgenden Zahlen in Ordnung ist
- Prüfe dann pro Zeile, ob die Zahlen entweder aufsteigend oder absteigend sind und ob die Differenz okay ist.

Lösungsvorschlag

```
private boolean istAufsteigend(int[] levels) {
    for (int i = 1; i < levels.length; i++) {
        if (levels[i] <= levels[i-1]) {// Gleichheit ist ein
Ausschlusskriterium!
            return false;
        }
    }
    return true;
}

private boolean istAbsteigend(int[] levels) {
    for (int i = 1; i < levels.length; i++) {
        if (levels[i] >= levels[i-1]) {// Gleichheit ist ein
Ausschlusskriterium!
            return false;
        }
    }
    return true;
}

private boolean distanzOkay(int[] levels) {
    for (int i = 1; i < levels.length; i++) {
        if (Math.abs(levels[i-1] - levels[i]) < 1 || Math.abs(levels[i-1] -
levels[i]) > 3) {
            return false;
        }
```

```
    }
    return true;
}

public void partOne() {
    int safeReports = 0;

    for (String line: inputLines) {
        // teile an den Leerzeichen auf
        String[] strLevels = line.split(" ");

        // deklariere & initialisiere ein int-Array
        int[] levels = new int[strLevels.length];
        // "übersetze" die einzelnen Strings in Integer
        for (int i = 0; i < strLevels.length; i++) {
            levels[i] = Integer.parseInt(strLevels[i]);
        }

        //prüfe Auf-/Absteigend und Distanz
        if ((istAufsteigend(levels) || istAbsteigend(levels)) &&
distanzOkay(levels)) {
            safeReports++;
        }

    }

    System.out.println(safeReports);
}
```

Teil 2

Für Teil 2 muss nur eine Kleinigkeit hinzugefügt werden!

Tipps zur Vorgehensweise

- Wir müssen der Reihe nach prüfen, ob ein Report safe wird, wenn die erste, zweite, dritte, … Zahl fehlt.
- Erstelle dazu eine neue Methode, die ein int-Array als ersten Parameter entgegennimmt und einen index als zweiten Parameter. Der zweite Parameter gibt an, welcher Index des Arrays entfernt werden soll. Die Methode muss also ein neues int-Array erstellen, das den übergebenen Index NICHT mehr enthält. Dieses neue Array soll dann zurückgegeben werden.
- Erweitere den Code aus Teil 1 so, dass du eine Schleife um die Prüfung auf Korrektheit baust: Du prüfst nun nie das "originale" Array, sondern du prüfst der Reihe nach: Wäre der report safe, wenn Index 1 fehlen würde, oder wenn Index 2 fehlen würde, oder …
- Sobald einer der "reduzierten" int-Arrays korrekt ist, kannst du mit `break` die umgebende Schleife frühzeitig abbrechen!

Lösungsvorschlag

```
private int[] reduceArray(int[] levels, int index) {
    // das neue int-Array muss eins kürzer sein
    int[] rLevels = new int[levels.length-1];

    // zum Speichern der aktuellen Einfügeposition im neuen Array
    int pos = 0;
    for (int i = 0; i < levels.length; i++) {
        if (i != index) {
            rLevels[pos] = levels[i];
            pos++;
        }
    }
    return rLevels;
}

public void partTwo() {
    int safeReports = 0;

    for (String line: inputLines) {
        // teile an den Leerzeichen auf
        String[] strLevels = line.split(" ");

        // deklariere & initialisiere ein int-Array
        int[] levels = new int[strLevels.length];
        // "übersetze" die einzelnen Strings in Integer
        for (int i = 0; i < strLevels.length; i++) {
            levels[i] = Integer.parseInt(strLevels[i]);
        }

        // erstelle pro Schleifendurchlauf ein reduziertes int-array, bei
dem jeweils ein Index fehlt
        for (int i = 0; i < levels.length; i++) {
            int[] reducedLevels = reduceArray(levels, i);
            //prüfe Auf-/Absteigend
            if ((istAufsteigend(reducedLevels) ||
istAbsteigend(reducedLevels)) && distanzOkay(reducedLevels)) {
                safeReports++;
                break;
            }
        }

    }
    System.out.println(safeReports);
}
```