Tag 13 - Point of Incidence

- Variante 1

Lösungshinweise Teil 1

- Lies die Patterns der Reihe nach ein und wandle sie zur einfacheren Verarbeitung in zweidimensionale boolean-Arrays um.
- Prüfe dann zunächst, ob an einer der Spalten die Spiegelung stattfindet. Lagere dies in eine Methode aus, die dir -1 zurückgibt, falls an keiner Spalte gespiegelt wird oder andernfalls die korrekte Spaltennummer.
    - Die Methode benötigt als Parameter das boolean-Pattern, die Breite und Höhe des patterns und die Anzahl der erlaubten Fehler in der Spiegelung. In Teil 1 ist KEIN Fehler in den Spiegelungen erlaubt (→ 0).
    - Prüfe der Reihe nach für x von links nach rechts (exklusive der letzten Spalte), ob diese Spalte die linke Spiegelachse ist (oder anders ausgedrückt: die Spalte direkt links der Spiegelachse).
    - Zähle pro möglicher x-Spiegelachsen-Position die Anzahl der vorhandenen Fehler.
    - Gehe pro möglicher x-Spiegelachsen-Position über alle Zeilen
    - Erhöhe der Reihe nach den Abstand (`diff`) von x, um zu prüfen, ob mit größer werdendem Abstand von der angenommenen Spiegelachse links und rechts derselbe Wert im pattern-Array steht. Sind die Zeichen nicht identisch, dann werden die Anzahl der Fehler erhöht. Sind mehr Fehler vorhanden als laut Parameter erlaubt, dann wird mit der nächsten angenommenen x-Spiegelachsen-Position weitergemacht.
    - Nur, wenn am Ende einer x-Spiegelachsen-Position die Fehleranzahl genau dem Parameter entspricht, dann wird die x-Position zurückgegeben (Achtung, den "bei-0-beginnenden-Index" anpassen). Sonst wird -1 zurückgegeben.
- Wenn bei den Spalten keine Spiegelachse gefunden wurde, dann wird entsprechend bei den Zeilen gesucht. Dazu muss eine zweite Methode exakt äquivalent aufgebaut werden.

Lösungsvorschlag

```
public int partOne() {
    int summe = 0;

    // speichert der Reihe nach immer die String-Zeilen EINES
Blocks/Patterns.
    ArrayList<String> patternStr = new ArrayList<String>();

    // zum prüfen, ob man bereits in der letzten Zeile ist
    int y = 0;

    // gehe von pattern zu pattern
    for (String line: inputLines) {
        y++;
        // wenn pattern endet oder letzte input-Zeile
        if (line.length() == 0 || y == inputLines.size()) {
```

```
            // füge auch noch die letzte Zeile der Datei zum Pattern hinzu.
            if (y == inputLines.size()) {
                System.out.println(line);
                patternStr.add(line);
            }

            int breite = patternStr.get(0).length();
            int hoehe = patternStr.size();

            // Wandle Strings eines patterns in ein boolean-array um
            boolean[][] pattern = stringsToArray(patternStr);

            // Prüfe zunächst alle Spalten
            int res = checkColumns(pattern, breite, hoehe, 0);
            if (res != -1) {
                summe += res;
            } else {
                res = checkRows(pattern, breite, hoehe, 0);
                summe += 100 * res;
            }

            // setze patternStr zurück
            patternStr = new ArrayList<String>();
            continue;
        }

        patternStr.add(line);
    }

    return summe;
}

private int checkRows(boolean[][] pattern, int breite, int hoehe, int
allowedErrors) {
    // prüfe jede mögliche horizontale Spiegelachse (y ist die obere zeile)
    for (int y = 0; y < hoehe - 1; y++) {
        // prüfe der reihe nach jede spalte
        int errors = 0;
        for (int x = 0; x < breite && errors <= allowedErrors; x++) {
            // prüfe von y nach oben + unten gehend
            int diff = 0;
            while (y-diff >= 0 && y+1+diff < hoehe) {
                if (pattern[x][y-diff] != pattern[x][y+1+diff]) {
                    errors++;
                    if (errors > allowedErrors) {
                        break;
                    }
                }
                diff++;
```

```
            }
        }
        // die Spalte wurde vollständig ohne Abbruch durchlaufen -> y war
korrekt (indexnummer anpassen)
        if (errors == allowedErrors) {
            return y + 1;
        }
    }
    return -1;
}

private int checkColumns(boolean[][] pattern, int breite, int hoehe, int
allowedErrors) {
    // prüfe jede mögliche vertikale Spiegelachse (x ist die linke spalte)
    for (int x = 0; x < breite - 1; x++) {
        // prüfe der reihe nach jede zeile
        int errors = 0;
        for (int y = 0; y < hoehe && errors <= allowedErrors; y++) {
            // prüfe von x nach links + rechts gehend
            int diff = 0;
            while (x-diff >= 0 && x+1+diff < breite) {
                if (pattern[x-diff][y] != pattern[x+1+diff][y]) {
                    errors++;
                    if (errors > allowedErrors) {
                        break;
                    }
                }
                diff++;
            }
        }
        // die Reihe wurde vollständig ohne Abbruch durchlaufen -> x war
korrekt (indexnummer anpassen)
        if (errors == allowedErrors) {
            return x + 1;
        }
    }

    return -1;
}

/**
 * Nimmt alle String-Zeilen eines patterns entgegen und wandelt sie in ein
 * zwei-dimensionales boolean-array um.
 */
private boolean[][] stringsToArray(ArrayList<String> lines) {
    boolean[][] pattern = new boolean[lines.get(0).length()][lines.size()];
    int y = 0;
    for (String l: lines) {
        for (int x = 0; x < l.length(); x++) {
            if (l.charAt(x) == '#') {
                pattern[x][y] = true;
```

```
                }
            }
            y++;
        }
        return pattern;
    }
```

Teil 2

Für Teil 2 muss nur eine Winzigkeit geändert werden: Es muss nun bei der korrekten Spiegelachse genau 1 Fehler auftreten!