Tag 1

Mehrere Zeilen werden eingelesen, diese bestehen aus Buchstaben und Zahlen. Man muss die erste Zahl und die letzte Zahl jeder Zeile finden. Es kann auch manchmal nur eine Zahl in der ganzen Zeile geben, dann fungiert diese sowohl als erste, als auch als letzte Zahl. Diese beiden Zahlen werden zusammengesetzt als zweistellige Zahl. Alle zweistelligen Zahlen aus jeder Zeile werden zusammenaddiert und bilden das gesuchte Ergebnis.

- Variante 1
- Variante 2

Hilfestellungen Variante 1

Teil 1

Tipp 1

Lies den Text zeilenweise und jede Zeile wiederum Zeichen für Zeichen. Prüfe für jeden einzelnen Character (Strings bestehen aus mehreren Charactern, in Java Typ char), ob dieser als Zahl interpretiert werden kann.

Tipp 2

Um den i-ten Character einer Zeile ("line") zu einem Integer-Wert zu "parsen" (zu übersetzen), kannst du folgenden Befehl verwenden:

```
int xxx = Integer.parseInt(line, i, i+1, 10);
```

Dabei gibt die 10 die Zahlenbasis an. Achtung: nicht jeder Character wird erfolgreich in eine int-Zahl umgewandelt werden können - es werden Exception (Ausnahmen/"Fehler") fliegen. Benutze daher z. B. folgende Schreibweise:

```
try {
   xxx = Integer.parseInt(line, i, i+1, 10);
} catch(Exception e) {
    // do nothing
}
```

Tipp 3

Erstelle 2 separate int-Variablen für die erste und zweite Zahl. Wenn du von vorne nach hinten jede Zeile durchgehst, dann wird die zweite Zahl in jedem Fall für jede neu gefundene int-Zahl aktualisiert. Die erste Zahl wird hingegen nur beim ersten Mal gesetzt. Da kannst du z. B. mit einer boolean prüfen, ob die erste Zahl bereits gesetzt wurde.

Lösungsvorschlag

```
int summe = 0;

for (String line : inputLines) {
    int zahl1 = 0;
    int zahl2 = 0;
    boolean firstNumberSet = false;

    for (int i = 0; i < line.length(); i++) {
        try {
            zahl2 = Integer.parseInt(line, i, i+1, 10);
            if (! firstNumberSet) {
                zahl1 = Integer.parseInt(line, i, i+1, 10);
                firstNumberSet = true;
            }
        } catch(Exception e) {
            // do nothing
        }
    }
    summe += zahl1*10 + zahl2;
}

return summe;
```

Teil 2

Der zweite Teil knüpft direkt an Teil 1 an. Von nun an müssen auch die ausgeschriebenen Zahlen ("one" für 1, "two" für 2, … bis 9) als Zahlen berücksichtigt werden. Dieser zweite Teil ist tatsächlich sehr knifflig, zumal es auch vorkommen kann, dass sich die Zahlen überlappen! So gibt es z. B. `oneight`, welches sowohl one als auch `eight`, also die 18 beinhaltet!

Wenn man den Code aus Teil 1 wiederverwenden möchte, so muss man also zuvor dafür sorgen, dass alle Text-Zahlen zu Ziffern-Zahlen übersetzt werden. Dies kann z. B. pro Zeile geschehen. Schreibe dies am besten in einer separaten Methode.

Tipp 1

Prüfe für jeden Index der Zeile (für jeden Charakter), ob an dieser Stelle entweder eine Ziffern-Zahl, oder eine ausgeschriebene Zahl steht. Wenn das der Fall ist, dann füge die entsprechende Ziffer hinten an einen (pro Zeile neuen) String an. Für diese Übersetzung kannst du tatsächlich denselben Code aus Teil 1 wiederverwenden. Prüfe also zunächst, ob der i-te Charakter als Zahl interpretiert werden kann. Falls nicht, dann landest du im catch-Block. Dort prüfst du nun der Reihe nach, ob ab dem i-ten Charakter der String mit einer ausgeschriebenen Zahl startet. Dazu kannst du `line.startsWith("one", i)` nutzen, wobei line die gesamte originale Zeile der Eingabe ist und i der aktuelle Index.

Lösungsvorschlag

```
/**
 * Schaue dir jeweils nur die ersten Zeichen an, ob dort eine
ausgeschriebene Zahl steht.
 * Füge dann nur diese erste ausgeschriebene Zahl in ein neues String-Array
an.
 */
private String textToNumbers(String line) {
    String lineAsNumbers = "";

    for (int i = 0; i < line.length(); i++) {
        int naechsteZahl;
        try {
            lineAsNumbers += Integer.parseInt(line, i, i+1, 10);
        } catch(Exception e) {
            /* Wenn man hier landet, dann ist das erste Zeichen KEINE
Ziffer, aber
             * vielleicht eine ausgeschriebene Zahl.
             */
            if (line.startsWith("one", i)) {
                lineAsNumbers += 1;
            } else if (line.startsWith("two", i)) {
                lineAsNumbers += 2;
            } else if (line.startsWith("three", i)) {
                lineAsNumbers += 3;
            } else if (line.startsWith("four", i)) {
                lineAsNumbers += 4;
            } else if (line.startsWith("five", i)) {
                lineAsNumbers += 5;
            } else if (line.startsWith("six", i)) {
                lineAsNumbers += 6;
            } else if (line.startsWith("seven", i)) {
                lineAsNumbers += 7;
            } else if (line.startsWith("eight", i)) {
                lineAsNumbers += 8;
            } else if (line.startsWith("nine", i)) {
                lineAsNumbers += 9;
            }
        }
    }

    return lineAsNumbers;
}

public int partTwo() {
    int summe=0;

    for (String line : inputLines) {
        String lineWithDigits = textToNumbers(line);
```

```
        int zahl1 = 0;
        int zahl2 = 0;
        boolean firstNumberSet = false;

        for (int i = 0; i < lineWithDigits.length(); i++) {
            try {
                zahl2 = Integer.parseInt(lineWithDigits, i, i+1, 10);
                if (! firstNumberSet) {
                    zahl1 = Integer.parseInt(lineWithDigits, i, i+1, 10);
                    firstNumberSet = true;
                }
            } catch(Exception e) {
                // do nothing
            }
        }
        System.out.println(lineWithDigits + " " + zahl1 + "" + zahl2);
        summe += zahl1*10 + zahl2;
    }

    return summe;
}
```

Hilfestellungen Variante 2

Hinweise zu Teil 1

Prinzipielles Vorgehen, um eine Zeile zu untersuchen:

- Wandle die Zeile in ein Array aus Zeichen um: `line.toCharArray();` Weitere Infos
- Betrachte jedes Zeichen in einer `foreach` Schleife oder einer ~~for~~ Schleife. ForEach-Schleife For-Schleife.
- Entscheide, ob das Zeichen eine Ziffer ist: `Character.isDigit(z)` isDigit
- Die erste gefundene Ziffer sind die Zehner, die letzte gefundene Ziffer sind die Einer. Mit einer Aggregationsvariablen kann man markieren, ob die 10er schon gefunden wurden. Wenn ja überschreibt man von diesem Zeitpunkt an die Einer mit der jeweils letzten gefundenen Ziffer. Sind Zehner und Einer dieselbe Ziffer, führt das hier nicht zu Problemen, weil zunächst der Marker auf "Zehner gefunden" gesetzt wird und deswegen dieselbe Ziffer als Einer gemerkt wird. Wenn in der Folge keine weitere Einerziffer gefunden wird, bleibt es dabei.

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

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:aoc:aco2023:day1:start?rev=1701452173

Last update: **01.12.2023 17:36**