Tag 10 - Pipe Maze

- Variante 1

Die heutige Aufgabe war nicht ganz einfach zu programmieren. Man musste viele Fälle einzeln durch if-else abdecken.

Hilfestellung Teil 1

- Wandle die Eingabe in ein zweidimensionales char-Array um. Speichere dieses am besten als Instanzvariable um flexibel darauf zugreifen zu können. Merke dir dabei direkt die Start-Koordinaten vom Buchstaben "S".
- Erstelle eine Hilfsmethode die du häufiger benötigen wirst: `getDirection(char from, char c)`. Die Methode soll uns sagen, in welcher Richtung man einen Buchstaben wieder verlässt, wenn man ihn aus einer bestimmten Richtung betritt. Du kannst dazu t, b, l, r nutzen für top, bottom, left, right. Bsp.: `getDirection(t, L)` gibt die Richtung `r` zurück. Du wirst für diese Methode zahlreiche Bedingungen überprüfen müssen. Denke auch an den Fall, dass man etwas falsches prüft - dann soll eine "Fehlerbuchstabe" zurückgegeben werden. Bsp.: `getDirection(l, L)` soll x zurückgeben.
- Prüfe nun für die Startkoordinate in alle vier Richtungen, ob diese Richtungen möglich wären, oder nicht (bei zweien wirst du bei der Richtungsüberprüfung den Fehlerbuchstaben x bekommen.
- Wiederhole nun bis du einmal im Kreis gelaufen bist und wieder bei "S" angelangt bist:
  - Je nach Richtung, in die du dich bewegst, änderst du deine x- oder y-Koordinate.
  - Lasse dir für diese neue Koordinate wieder die nächste Richtung mithilfe der oberen Methode geben.
  - Pro Schritt erhöhst du den Schrittzähler.
- Am Ende ist das Ergebnis die Hälfte des Schrittzählers.

Lösungsvorschlag

```
/**
 * Man kommt aus Richtung "from" (l(eft), r(ight), t(op), b(bottom))
 * zum Buchstaben "c"
 * und bekommt die neue Richtung als Rückgabewert zurück (x für falsche
Eingaberichtung).
 */
private char getDirection(char from, char c) {
    if (c == 'J') {
        if (from == 'l') {
            return 't';
        } else if (from == 't') {
            return 'l';
        }
    } else if (c == '7') {
        if (from == 'l') {
            return 'b';
        } else if (from == 'b') {
            return 'l';
```

```
        }
    } else if (c == 'F') {
        if (from == 'b') {
            return 'r';
        } else if (from == 'r') {
            return 'b';
        }
    } else if (c == 'L') {
        if (from == 't') {
            return 'r';
        } else if (from == 'r') {
            return 't';
        }
    } else if (c == '|') {
        if (from == 't') {
            return 'b';
        } else if (from == 'b') {
            return 't';
        }
    } else if (c == '-') {
        if (from == 'l') {
            return 'r';
        } else if (from == 'r') {
            return 'l';
        }
    }

    return 'x';
}

private ArrayList<Character> getStartDirections(int x, int y) {
    ArrayList<Character> dirs = new ArrayList<Character>();

    if (y > 0) {
        if (getDirection('b', welt[x][y-1]) != 'x') {
            dirs.add('t');
        }
    }

    if (x < inputLines.get(0).length() - 1) {
        if (getDirection('l', welt[x+1][y]) != 'x') {
            dirs.add('r');
        }
    }

    if (y < inputLines.size() - 1) {
        if (getDirection('t', welt[x][y+1]) != 'x') {
            dirs.add('b');
        }
    }
```

```
    if (x > 0) {
        if (getDirection('r', welt[x-1][y]) != 'x') {
            dirs.add('l');
        }
    }

    return dirs;
}

public int partOne() {

    welt = new char[inputLines.get(0).length()][inputLines.size()];

    int nextX = 0;
    int nextY = 0;

    for (int y = 0; y < inputLines.size(); y++) {
        String line = inputLines.get(y);
        char[] chars = line.toCharArray();
        for (int x = 0; x < chars.length; x++) {
            if (chars[x] == 'S') {
                nextX = x;
                nextY = y;
            }
            welt[x][y] = chars[x];
        }
    }

    // finde die Startrichtung
    char nextDir = getStartDirections(nextX, nextY).get(0);

    int steps = 0;
    do {

        if (nextDir == 't') {
            nextY--;
            nextDir = getDirection('b', welt[nextX][nextY]);
        } else if (nextDir == 'r') {
            nextX++;
            nextDir = getDirection('l', welt[nextX][nextY]);
        } else if (nextDir == 'b') {
            nextY++;
            nextDir = getDirection('t', welt[nextX][nextY]);
        } else if (nextDir == 'l') {
            nextX--;
            nextDir = getDirection('r', welt[nextX][nextY]);
        } else {
            // Zum Fehler ausgeben (was nicht vorkommen sollte)
            System.out.println(nextDir + " " + nextX + " " + nextY);
        }
```

```
        steps++;
    } while (welt[nextX][nextY] != 'S');

    return (steps + 1) / 2;
}
```

Hilfestellung Teil 2

Teil 2 ist recht knifflig und benötigt die verzwickte Überprüfung von Bedingungen.

- Erstelle gleich zu Beginn ein weiteres boolean[][] Array, in welchem du später beim Durchlaufen des Loops alle Koordinaten auf true setzt, die dem gelaufenen Pfad entsprechen.
- Nachdem du die Startkoordinaten von "S" gefunden hast musst du "S" durch den korrekten Character ersetzen der an diese Stelle gesetzt werden kann (L, J, F, 7, | oder -).
- Wenn du den ganzen Loop durchlaufen hast, dann musst du beginnen sämtliche Character nochmals zeilenweise zu betrachten.
    - Speichere dir pro Zeile in einer boolean-Variablen, ob du gerade *inside* oder *outside* bist (im geschlossenen Kreis oder außerhalb).
    - Wenn du aktuell auf einer Koordinate bist, die du im boolean-Array **nicht** als Pfad markiert hast, dann wird dein inside-Zähler um eins erhöht, falls deine eben erstellte boolean-Variable anzeigt, dass du *inside* bist. Sonst gehst du mit `continue` einfach zur nächsten Koordinate.
    - Andernfalls (du bist auf einem Pfad) musst je nach Character c umfangreich unterscheiden:
        - Wenn c der Pipe-Operator | ist, dann kannst du direkt die inside/outside-Variable zum jeweils anderen Wert switchen, weil du weißt, dass durch das Passieren dieses Zeichens auf jeden Fall von innen nach außen gelangt bist oder andersherum.
        - Wenn c nun L oder F ist, dann bedeutet das, dass danach eine längere Sequenz beginnt, die horizontal weiterläuft, ohne dass zwangsweise sofort ein Wechseln von innen nach außen oder andersherum stattfindet. Z. B.: L—7, LJ oder F-7. Sehr wichtig ist an dieser Stelle aber, dass du dir diesen Anfangsbuchstaben der horizontalen Sequenz (L oder F) merkst.
        - Wenn c ein horizontales Bindestrich - ist, dann kannst du einfach mit `continue` weitermachen.
        - Wenn c das eine mögliche Ende einer horizontalen Sequenz anzeigt (J), dann musst du unterscheiden, was das Zeichen vom Beginn der horizontalen Sequenz war.
            - Wenn die Sequenz mit L begann, dann heißt das, dass der Pfad **nicht** deinen Weg gekreuzt hat (z. B. L—J oder LJ) und du damit auch nicht von inside nach outside oder anders herum gewechselt hast. Du brauchst gar nichts tun (`continue`).
            - Wenn die Sequenz hingegen mit F begonnen hat, dann hat der Pfad deinen Weg gekreuzt (z. B. FJ oder F—J). Inside/outside muss also getauscht werden.
        - Wenn c das andere mögliche Ende einer horizontalen Sequenz anzeigt (7), dann musst du entsprechend andersherum verfahren.

Lösungsvorschlag

```
public int partTwo() {
```

```
    welt = new char[inputLines.get(0).length()][inputLines.size()];
    boolean[][] pfad = new
boolean[inputLines.get(0).length()][inputLines.size()];

    int nextX = 0;
    int nextY = 0;

    for (int y = 0; y < inputLines.size(); y++) {
        String line = inputLines.get(y);
        char[] chars = line.toCharArray();
        for (int x = 0; x < chars.length; x++) {
            if (chars[x] == 'S') {
                nextX = x;
                nextY = y;
            }
            welt[x][y] = chars[x];
        }
    }

    // finde die Startrichtungen
    ArrayList<Character> startDirs = getStartDirections(nextX, nextY);
    char nextDir = startDirs.get(0);

    // ersetzte 'S' durch korrektes Zeichen
    if (startDirs.contains('l') && startDirs.contains('t')) {
        welt[nextX][nextY] = 'J';
    } else if (startDirs.contains('t') && startDirs.contains('r')) {
        welt[nextX][nextY] = 'L';
    } else if (startDirs.contains('r') && startDirs.contains('b')) {
        welt[nextX][nextY] = 'F';
    } else if (startDirs.contains('b') && startDirs.contains('l')) {
        welt[nextX][nextY] = '7';
    } else if (startDirs.contains('t') && startDirs.contains('b')) {
        welt[nextX][nextY] = '|';
    } else if (startDirs.contains('l') && startDirs.contains('r')) {
        welt[nextX][nextY] = '-';
    }

    int startX = nextX;
    int startY = nextY;

    int steps = 1;
    do {
        pfad[nextX][nextY] = true;
        if (nextDir == 't') {
            nextY--;
            nextDir = getDirection('b', welt[nextX][nextY]);
        } else if (nextDir == 'r') {
            nextX++;
            nextDir = getDirection('l', welt[nextX][nextY]);
        } else if (nextDir == 'b') {
```

```
            nextY++;
            nextDir = getDirection('t', welt[nextX][nextY]);
        } else if (nextDir == 'l') {
            nextX--;
            nextDir = getDirection('r', welt[nextX][nextY]);
        } else {
            // Zum Fehler ausgeben
            System.out.println(nextDir + " " + nextX + " " + nextY);
        }
    } while (nextX != startX || nextY != startY);

    int countInside = 0;
    for (int y = 0; y < inputLines.size(); y++) {
        boolean outside = true;
        char pathBegin = 'x';
        for (int x = 0; x < inputLines.get(0).length(); x++) {
            if (!pfad[x][y]) {
                if (outside) {
                    continue;
                } else {
                    countInside++;
                }
            } else {
                char c = welt[x][y];
                if (c == '|') {
                    outside = !outside;
                } else if (c == 'L' || c == 'F') {
                    pathBegin = c;
                } else if (c == '-') {
                    continue;
                } else if (c == 'J') {
                    if (pathBegin == 'L') {
                        continue;
                    } else if (pathBegin == 'F') {
                        outside = !outside;
                    }
                } else if (c == '7') {
                    if (pathBegin == 'L') {
                        outside = !outside;
                    } else if (pathBegin == 'F') {
                        continue;
                    }
                }
            }
        }
    }

    return countInside;
}
```

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

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:aoc:aco2023:day10:start?rev=1702229909

Last update: **10.12.2023 17:38**