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;
}
```

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

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:aoc:aco2023:day10:start?rev=1702228745

Last update: **10.12.2023 17:19**