Day 14: Restroom Redoubt

Teil 1

Obwohl die Beschreibung des Problems ein zweidimensionales Array nahelegt, kann man den ersten Teil in einem einzigen Schleifendurchlauf, direkt während des Einlesens jeder einzelnen Zeile, lösen.

Grundidee:

- Pro Iteration bewegt sich ein Roboter um eine bestimmte Strecke in x- und y-Richtung. Wenn dabei in einer (oder beiden) Richtungen die Grenze über/-unterschritten wird, dann muss der Roboter wieder von der anderen Seite hineinlaufen. Das ist eine klassische `modulo`-Aufgabe. Die erlaubte Koordinate ist also die berechnete Koordinate `modulo` breite/höhe. Damit dies auch mit negativen Werten funktioniert nutzen wir hier nicht den %-Befehl, sondern die Methode `Math.floorMod(berechneteKoordinate, breite bzw. höhe)`.
- Wir können aber ungemein Zeit sparen. Für den Modulo ist es egal, ob der Roboter nur ein bisschen, oder ganz stark aus der Grenze hinausgelaufen ist. Wir können also direkt alle 100 Iterationen in einem Schritt berechnen und anschließend den oberen Modulo-Befehl einmal ausführen!
- Anschließend können wir noch pro Roboter direkt prüfen, in welchem Quadrant er nun steht und eine entsprechende Variable erhöhen.
- Damit sind alle Berechnungen zu diesem Roboter direkt abgeschlossen, wir können die nächste Zeile vom Input einlesen und müssen nichts mehr zum aktuellen Roboter speichern.

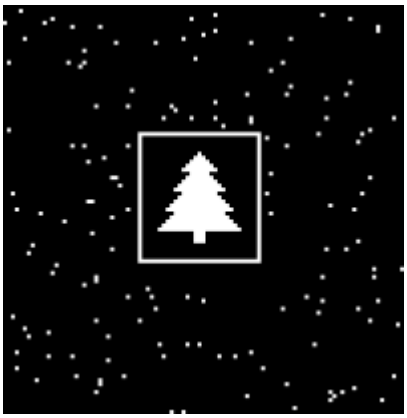
Lösungsvorschlag

```
public void partOne() {
    int[] quadrants = new int[4]; // zum Speichern der Roboteranzahlen in
den Quadranten
    for (String line: inputLines) {
        int[] robot = new int[4];
        robot[0] = Integer.parseInt(line.split(",")[0].split("=")[1]); // x
        robot[1] = Integer.parseInt(line.split(",")[1].split(" ")[0]); // y
        robot[2] = Integer.parseInt(line.split(",")[1].split("=")[1]); // vx
        robot[3] = Integer.parseInt(line.split(",")[2].trim()); // vy

        // 100 mal bewegen und davon den modulo berechnen
        robot[0] = Math.floorMod(robot[0]+(100*robot[2]), width);
        robot[1] = Math.floorMod(robot[1]+(100*robot[3]), height);

        if (robot[0] < width/2 && robot[1] < height/2) {
            quadrants[0]++;
        } else if (robot[0] > width/2 && robot[1] < height/2) {
            quadrants[1]++;
        } else if (robot[0] < width/2 && robot[1] > height/2) {
            quadrants[2]++;
        } else if (robot[0] > width/2 && robot[1] > height/2) {
            quadrants[3]++;
```

```
        }
    }
    System.out.println(quadrants[0]*quadrants[1]*quadrants[2]*quadrants[3]);
}
```

Teil 2

Wider Erwarten muss man für Teil 2 nun doch schrittweise vorgehen und alle Roboter in einer ArrayList speichern. Übrigens: Das erwartete Ergebnis sollte ungefähr so aussehen wie das rechte Bild .



Man muss nun in vielen Iterationen alle Roboter jeweils eine einzige Bewegung durchführen lassen. Nach jedem Durchlauf muss man überprüfen, ob die neuen Koordinaten der Roboter ein Bild der gezeigten Form ergeben - ja, dies erfordert viel manuelle Handarbeit! Nein, man kann dies (ohne eine KI-Bilderkennung) nicht automatisieren. Da damit zu rechnen ist, dass sehr viele Bilder/Iterationen durchsucht werden müssen (4-stelliger Bereich), bietet es sich an, richtige Bilder in Form von PNGs zu erstellen. Deren Thumbnails können im Dateimanager schneller durchsucht werden als entsprechend viele monospace-"Bilder" im Terminal.

Vorgehensweise:

- Zur Erstellung der PNGs müssen folgende Bibliotheken importiert werden:
  - `import javax.imageio.ImageIO;`
  - `import java.awt.image.BufferedImage;`
  - `import java.io.File;`
- Speichere alle Roboter-Werte in einer ArrayList.
- Wiederhole bis zu 10.000 Mal:
  - Berechne für jeden Roboter ein einzelne Bewegung.
  - Speichere die Koordinaten in einem zweidimensionalen char-Array. Setze z. B. '#' für einen Roboter.
  - Übertrage das char-Array in ein Bild-Objekt und speichere dieses als PNG-Datei.
- Durchsuche anschließend als Thumbnails alle Bilder bis du das richtige gefunden hast. Die Antwort ist die Bildnummer (wobei du darauf achten musst, ob du noch +1 dazuaddieren musst, falls deine Schleife mit i=0 angefangen hat).

Lösungsvorschlag

```
public void partTwo() {
    ArrayList<int[]> robots = new ArrayList(); // zum Speichern aller
Roboter
```

```
    for (String line: inputLines) {
        int[] robot = new int[4];
        robot[0] = Integer.parseInt(line.split(",")[0].split("=")[1]); // x
        robot[1] = Integer.parseInt(line.split(",")[1].split(" ")[0]); // y
        robot[2] = Integer.parseInt(line.split(",")[1].split("=")[1]); // vx
        robot[3] = Integer.parseInt(line.split(",")[2].trim()); // vy

        robots.add(robot); // zur ArrayList hinzufügen
    }

    // Der umgebende try-catch-Block ist nötig, da beim Datei-Schreiben
Fehler auftreten können und diese abgefangen werden müssen.
    try {
        // führe maximal 10.000 Iterationen durch - das sollte genügen
        for (int i = 0; i < 10000; i++) {
            char[][] pic = new char[width][height]; // speichert das
entstehende Bild zunächst in einem char-Array
            for (int[] robot: robots) {
                robot[0] = Math.floorMod(robot[0]+robot[2], width);
                robot[1] = Math.floorMod(robot[1]+robot[3], height);
                pic[robot[0]][robot[1]] = '#'; // markiert die Stelle an der
der Roboter nach einer Iteration ist
            }

            // Erstelle ein Bild-Objekt
            BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_BYTE_BINARY);

            // Übertrage jedes Pixel vom char-Array in das Bild-Objekt
            for (int y = 0; y < height; y++) {
                for (int x = 0; x < width; x++) {
                    // Der letzte Parameter ist ein inline-if: wenn ein '#'
gespeichert ist, dann
                    // wird der Farbcode FFFFFF gesetzt, andernfalls 000000.
                    image.setRGB(x, y, pic[x][y] == '#' ? 0xFFFFFF :
0x000000);
                }
            }
            // Setze als Dateipfad den relativen Unterordner trees und den
Dateinamen trees123.png (für das Beispiel der 123-ten Iteration).
            File output = new File("trees/tree" + i + ".png");

            // Schreibe die Datei
            ImageIO.write(image, "png", output);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```