Aufgabe 3

"In dem Tar-File strecken.tgz befinden sich Dateien mit jeweils 4 Koordinaten pro Zeile. Diese stellen jeweils die x- und y-Koordinaten eines Start- und Endpunktes einer Strecke dar. Lesen Sie die Datei Strecken.dat ein und ermitteln Sie die Anzahl der sich schneidenden (d.h. mindestens ein gemeinsamer Punkt) Strecken, indem Sie jedes Paar von Strecken gegeneinander testen. Messen Sie die pro Datei aufgewendete Zeit."

Bei dieser Aufgabe gilt es mit Hilfe des Line Sweep Algorithmus herauszufinden, wieviele Schnittpunkte zwischen den Linien, die in den Files definiert sind, auftreten.

Dabei kommt der aus Aufgabe 1 bekannte CCW zum Bestimmen von Schnittpunkten zwischen Segmenten zum Einsatz.

Implementierung

1) Hauptroutine zum Einlesen der 3 Streckendateien

```
// For every file calculate number of intersections
for (String path : Arrays.asList(
       PATH_STRECKEN_1_000, PATH_STRECKEN_10_000, PATH_STRECKEN_100_000))
       {
              System.out.println("Starting lineSweep for " + path);
              System.out.println("# of intersections: " + numIntersectionsForFile(path));
       }
```

2) Lesen der Segmente aus Datei in eine Liste

```
private static List<Segment> readLines(String path) throws IOException {
       return Files.lines(Paths.get(path)).map((line) ->
       byLine(line)).collect(Collectors.toList());
}
```

3) Initialisieren der EventQueue

```
Queue<Event> eventQueue = new PriorityBlockingQueue<>();
       segments.forEach((segment) -> {
            eventQueue.add(new SegmentStart(segment));
            eventQueue.add(new SegmentEnd(segment));
       });
```

4) Verarbeiten der EventQueue durch Line Sweep

```
while ((event = eventQueue.poll()) != null) {
    currentEventX = event.getX();
    switch (event.getType()) {
    case SEGMENT_START:
        Segment segment = event.getSegment();
        sweepLineSegments.add(segment);
        Collections.sort(sweepLineSegments);
        Collection<Segment> neighbours = findNeighbours(segment);
        for (Segment neighbourSegment : neighbours) {
            if (segment.intersectsLine(neighbourSegment)) {
                Point2D intersection = getIntersection(segment, neighbourSegment);
                eventQueue.add(new SegmentsIntersect(segment, neighbourSegment,
intersection.getX()));
            }
        }
        break;
    case SEGMENT_END:
        sweepLineSegments.remove(event.getSegment());
    case SEGMENTS_INTERSECT:
        numIntersections++;
    default:
        break;
    }
}
```

Ergebnis

```
Starting lineSweep for resources/Strecken_1000.txt
# of intersections: 1810
Starting lineSweep for resources/Strecken_10000.txt
# of intersections: 18221
Starting lineSweep for resources/Strecken_100000.txt
# of intersections: 181930
```