Europaschule Schulzentrum SII Utbremen

Programmier-Praktikum

Routenplaner

vorgelegt von...

Namen:	Jesse Strathmann, Nele Hugo
Klasse:	DQI17
Abgabe:	14.06.20

Inhaltsverzeichnis

1. Projektbeschreibung

Die Aufgabe war es, ein Programm in Java mit einer ansprechenden grafischen Oberfläche zu entwickeln, mit dem man eine Route durch ein Autobahnnetz in Deutschland planen kann.

Der Benutzer soll hierzu die Möglichkeit haben, einen Startpunkt und ein Endpunkt in der Oberfläche auswählen zu können und basierend darauf dann eine Route mit allen Zwischenzielen auf seiner Reise zu erhalten. Diese Route soll in einem Fenster innerhalb des Programms angezeigt werden

Die Daten für die Autobahnverbindungen wurden in einer XML-Datei bereit gestellt und sollen vom Programm eingelesen werden. Außerdem sollen die Knotenpunkte auf einer Deutschlandkarte, welche ebenfalls bereitgestellt wurde, angezeigt werden. Bei einer geplanten Route, sollen die Knoten und Kanten visuell hervorgehoben werden.

Die gewünschte Route ist hierbei immer der kürzeste Weg zwischen Start und Ziel.

Bonuspunkte gibt es, wenn die Suche von Start- und Endknoten dynamisch erfolgt und wenn die berechnete Route in eine Textdatei exportiert werden kann.

2. Theoretische Überlegungen

Zunächst war zu klären, wie wir die Daten aus der XML-Datei auslesen. Hierzu entscheiden wir uns für die im JDK integrierte Lösung eines XML-Parsers, mit dem wir ohne viel Aufwand die Daten extrahieren können. Komplementär dazu entwerfen wir basierend auf den bereitgestellten Daten Modellklassen, welche die extrahierten Daten in einer für uns leicht verwendbaren Form speichern und zugänglich machen.

Für die Benutzeroberfläche greifen wir relativ alternativlos auf das JavaFX-Framework zurück, da es das ausgereifteste Java-Framework für moderne Benutzeroberflächen ist und wir ohnehin schon in vergangenen Projekten damit gearbeitet haben.
Für den Aufbau der grafischen Oberfläche gibt es durch die Aufgabenstellung wenig Spielraum für die Auswahl an Komponenten. Entscheiden mussten wir uns jedoch für eine Vorgehensweise um das Bild der Deutschlandkarte in der Oberfläche anzuzeigen und im Nachhinnein auch durch das Programm auf dieser Karte zu zeichnen, um Knoten und Kanten darzustellen. Hier bietet sich ein Canvas an, da mit einem Canvas zum einen Bilder gezeichnet werden können und zum anderen die Knoten und Kanten durch die entsprechende Methoden. Eine Alternative wäre ein Stackpane gewesen, wofür wir uns jedoch nicht entschieden haben.

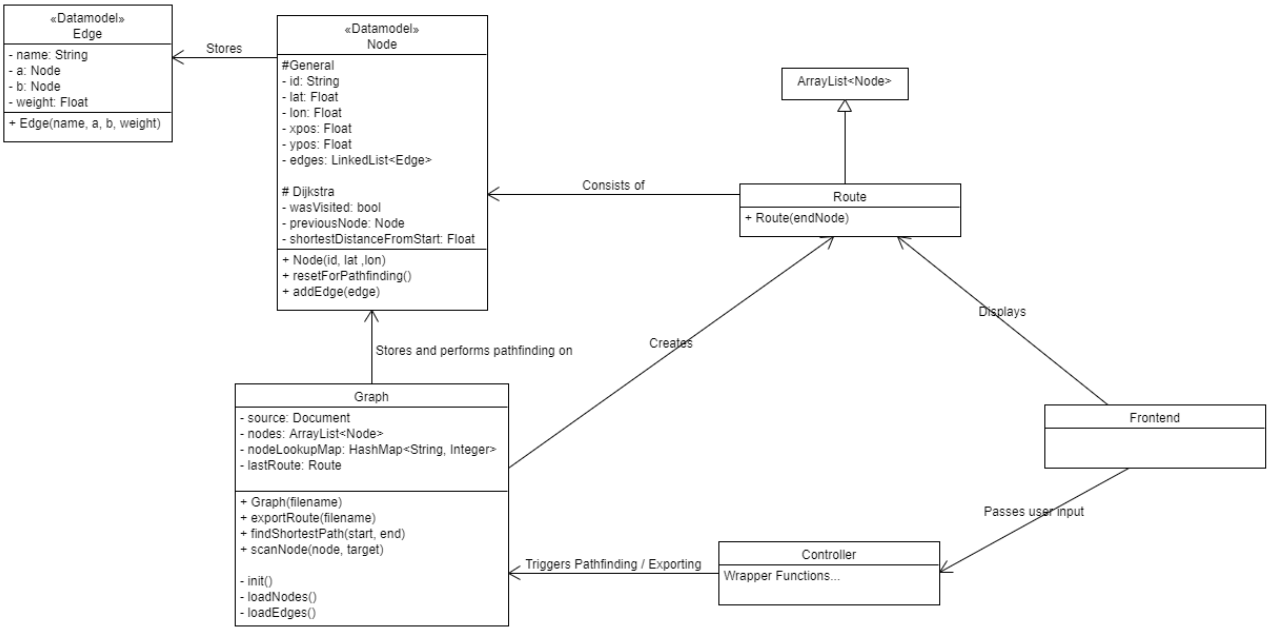
Für Umrechnung von Latitude und Longitude in die korrekten X- und Y-Koordinaten auf unserem Canvas beschlossen wir, zwei Referenzpunkte aus dem Datensatz zu entnehmen und diese per Google-Maps-Suche lokalisieren und durch Handarbeit in

unserer Karte eintragen um so die Anzahl an Pixel pro Latitude bzw. Longitude zu erhalten durch die Bildung der Differenzen unserer beider Referenzpunkte.

Die nächste Überlegung die wir treffen mussten, war die Implementation des Dijkstra-Algorithmus. Hierzu definierten wir Eigenschaften, um die wir die Modellklasse eines Knoten erweitern müssen, damit der Dijkstra-Algorithmus implementiert werden kann. Ein Beispiel für eine solche Eigenschaft ist der (aktuelle) Abstand zum Startknoten, welcher durch Berechnungen des Dijkstra-Algorithmus bestimmt und bei Bedarf geändert wird.

Auch die Speicherung der ausgelesenen Daten bzw. der Modellklassen bedarf der Überlegung. Hierfür entscheiden wir uns, in jedem Knoten, die von ihm ausgehenden Verbindungen, in einer Liste zu speichern. Die Knoten wiederum sind ebenfalls in einer Liste gespeichert, die den Zugriff auf diese ermöglicht. Um auch von einer ID eines Knotens auf den Index in der Liste schließen zu können und das in einer möglichst kurzen Zeit, speichern wir die ID eines Knotens mit dem dazugehörigen Index in einer HashMap, da diese bei Zugriffsoperationen im Normallfall eine Laufzeitkomplexität von O(1) haben.

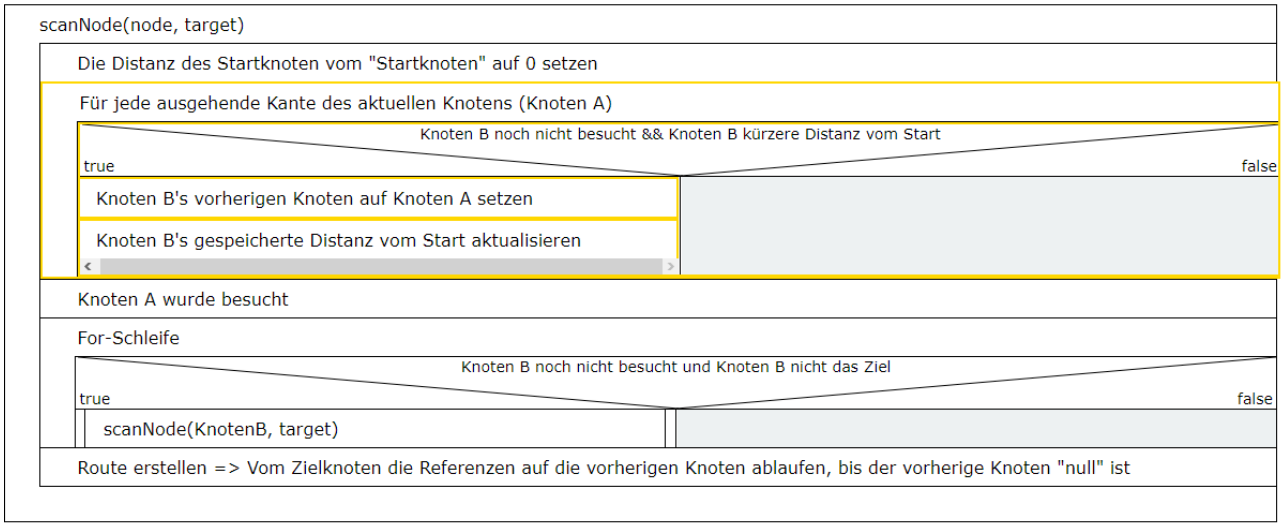
3. Softwareentwurf



(im img-Ordner zu finden)

Für unseren Softwareentwurf entschieden wir uns zunächst ein Klassendiagramm zu erstellen, um einen ersten Überblick über den Umfang des Projektes zu erhalten und um im weiteren Verlauf eine gemeinsame Basis für unsere Arbeitsteilung und Zielvorstellungen haben. Das Klassendiagramm besteht aus zwei Modellklassen, wie wir sie in der theoretischen Überlegung angebracht haben, einer extra Modellklasse für die berechneten Routen, eine Graphen-klasse in der die Speicherung der Modellklassen aus unserer theoretischer Überlegung einfloss, und einen Controller, der der grafischen Oberfläche Zugriff auf die im Graphen gebündelte Logik gewährt.

Im Konstruktor der Graphen-Klasse wird der Name des XML-Dokuments übergeben, aus welcher dann die Daten gelesen werden und die Modellklassen instanziiert und diese Instanzen dann gespeichert werden. Der Dijkstra-Algorithmus ist hier als „scanNode(node, target)“ festgehalten. Den Algorithmus haben wir in einem Struktogramm geplant:

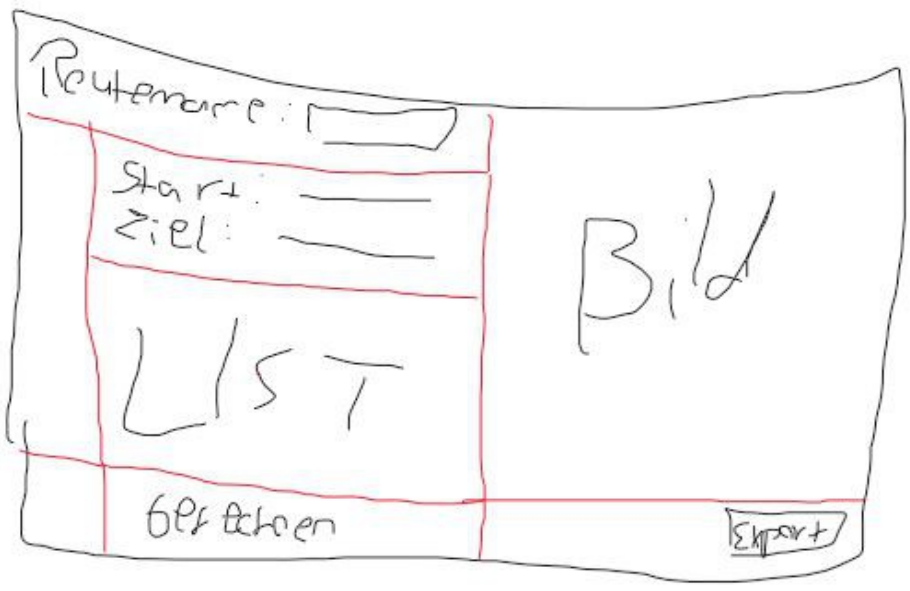


(im img-Ordner zu finden)

Knoten A bezeichnet hierbei den aktuellen Knoten und Knoten B den Knoten am Ende der aktuellen Verbindung.

Dieser Algorithmus basiert auf Rekursion, was bei einer enorm großen Anzahl an Knoten eventuell durch einen Stackoverflow zu Programmabstürzen führen kann. In unserem Fall stellt das jedoch kein Problem dar.

Nun zur GUI: Hier haben wir angefangen eine grobe Skizze anzufertigen.



Hierbei haben wir sofort in der Skizze schon abgegrenzt, wie dies ungefähr organisiert sein soll, mithilfe von roten Linien. Anhand der roten Linien konnte man sehr leicht anfangen durchzuplanen, wo genau eine HBox oder eine VBox benötigt werden könnte. Noch einmal kurz zur Erklärung: Eine HBox verläuft horizontal und eine VBox vertikal. Beispielsweise wurde als aller erstes eine große HBox erstellt, die beide Seiten (links und rechts) quasi abgrenzt. Während auf der rechten Seite in der HBox nur das Bild von der Karte und ein Export Button ist, sollte auf der linken Seite der Routenname stehen, die Start- und Zielbestimmung, eine Liste, sowie ein Button zum Berechnen.

4. Projektplanung

4.1 Persönliche Zielsetzungen

Jesse: Mein persönliches Ziel in diesem Projekt ist es, den Dijkstra-Algorithmus zu verstehen und somit auch implementieren zu können. Ich denke, dass es nützlich sein kann, wenn man sich mehr Wissen aneignet – vor allem auch in Themenbereichen, mit denen man zuvor nicht sehr viel zu tun hatte. Demnach ist es mal eine gute Abwechslung, sich mit Routenplanungen zu beschäftigen.

Nele: Ich bin nicht sehr begeistert vom Projekt – im Sinne von, mir würde keine wirkliche Zielsetzung für dieses Projekt einfallen. In der Zukunft plane ich nicht wirklich mit Routenplanungen oder Ähnlichem zu tun zu haben, weshalb ich nicht so interessiert bin. Das ist vermutlich das erste Projekt, wo ich keine wirkliche Zielsetzung auf das Thema bezogen habe. Demnach möchte ich mich in diesem Projekt wieder der GUI widmen → allerdings mit dem Unterschied, dass ich den Code dieses Mal besser organisieren möchte, als in meinen vorherigen Projekten – es gab nämlich Momente, wo ich meinen gesamten Code in nur eine Methode geschrieben habe.

4.2 Arbeitsaufteilung

Wenn man sich unsere Zielsetzungen anschaut, ist die Arbeitsaufteilung ziemlich klar. Das gute an unserer Zusammenarbeit ist, dass unsere Interessen sich unterscheiden. Während eine Person eher Lust auf das Frontend hat, hat die andere Person eher Lust auf das Backend. Demnach hat Nele sich um die GUI gekümmert und Jesse sich um das, was dahinter steckt.

5 Projektverlauf

5.1 Überblick

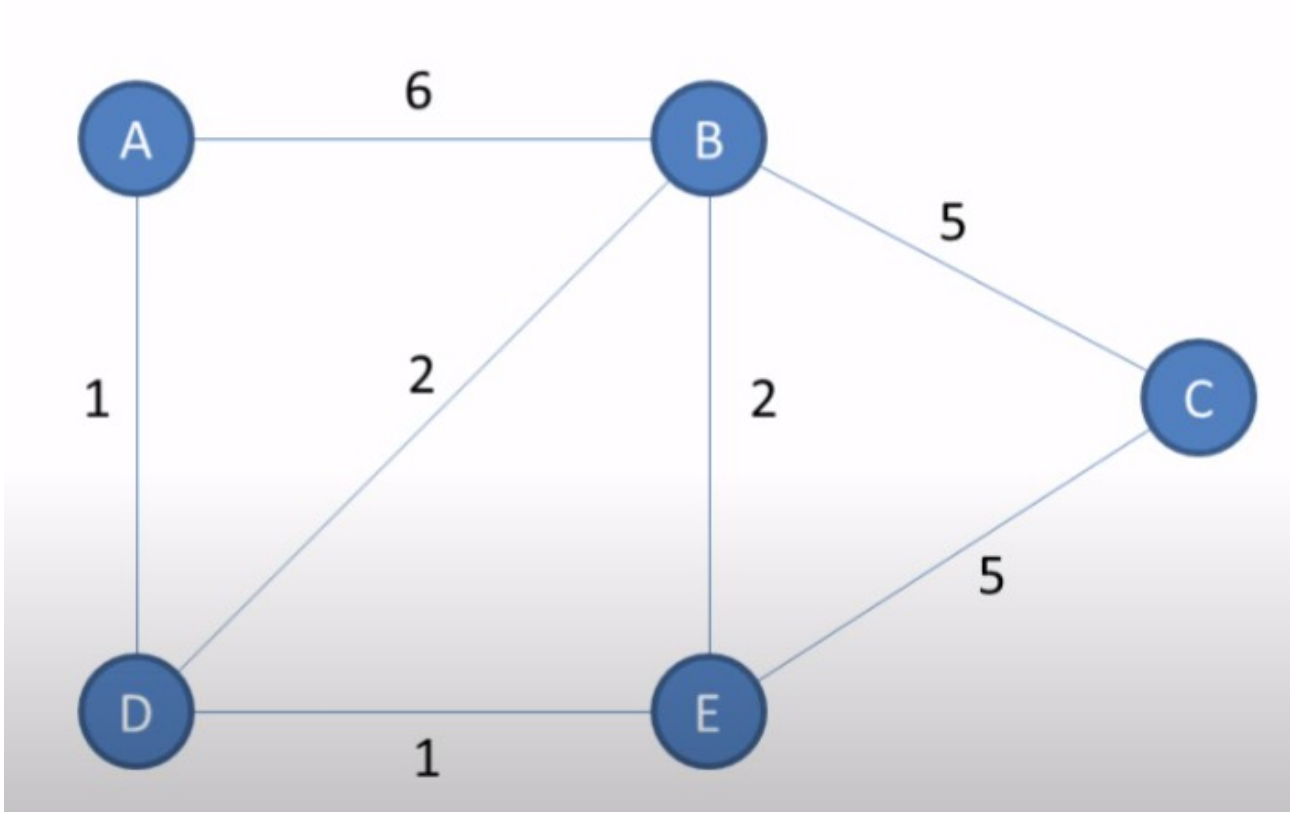
Selbst wenn der PP Unterricht in der Schule stattgefunden hätte, wären wir nicht in der gleichen Halbgruppe gewesen – aus dem Grund stand Corona diesem Projekt nicht

wirklich im Weg. Wir konnten während des Projektes gut miteinander kommunizieren und haben neben unsere Kommunikationssoftware – Discord – auch GitHub aktiv genutzt, um unseren Fortschritt zu teilen.

Während einer angefangen hat, sich mit dem Dijkstra-Algorithmus zu beschäftigen, hat das andere Gruppenmitglied angefangen sich darum zu kümmern, was genau auf der GUI angezeigt werden muss. Hierbei wurden ggf. Nachfragen durchgeführt, was genau benötigt wird.

5.2 Testprotokolle

- Im Vordergrund unserer Tests stand der Dijkstra-Algorithmus, da dieser der Funktionalität der Anwendung zugrunde liegt. Hierzu erstellten wir einen Graphen zum Testen:



(Quelle: https://www.youtube.com/watch?v=pVfj6mxhdMw)

Anhand dieses Graphen und der dazugehörigen Kontrolle durch das angegebene Video haben wir den Algorithmus implementiert und anschließend mit den Beispielen im Video getestet.

- Ein weiterer Schwerpunkt unserer Tests war das korrekte Anzeigen der Knoten und Kanten auf der Deutschlandkarte. Hierzu prüften wir, ob unsere berechneten Punkte in etwa mit der Position, die auf Google-Maps gezeigt wird, übereinstimmen. Bei diesen Tests gab es keine fehlerfreien Testmöglichkeiten, wir mussten uns hier auf Abschätzungen verlassen.

- Des Weiteren waren noch die üblichen Tests der grafischen Oberfläche und der Benutzereingaben zu prüfen. Beispielhaft dafür wäre das Korrekte Anzeigen der grafischen Oberfläche, also ohne Überlappungen oder Designbrüchen. Auch ob die Eingabe von leeren Start- und Endfeldern korrekt abgefangen wird, musste sichergestellt werden.

- Auch das Exportieren der berechneten Route in eine Textdatei haben wir getestet. Hier war es uns wichtig, dass das Exportieren mit nur einem Knopfdruck funktioniert. Zu testen war, die Erstellung der Textdatei mit dem korrekten Namen, bzw. dem Platzhalternamen, falls der Benutzer keinen gewünschten Namen eingetragen hat und auch ob der Inhalt dieser Textdatei mit der in der Oberfläche angezeigten Route übereinstimmt. Hinzu kommt, ob das Überschreiben einer bereits vorhandenen Textdatei fehlerfrei abläuft, also ohne dass der Inhalt unlesbar wird oder einfach nur an den vorhandenen Inhalt angehangen wird.

6 Projektergebnisse

6.1 Ergebnisse

Wir haben erfolgreich eine Routenplaner Software entwickelt, mit der es möglich ist, Routen zu berechnen. Sobald man das Programm startet, öffnet sich ein Fenster mit verschiedenen Elementen. Zum einen, hat man die Möglichkeit für die Route einen Namen festzulegen, in dem Eingabefeld hinter dem Titel „Routenname“. Anschließend kann man einen Start, sowie einen Zielort mithilfe eines Dropdown Menü's auswählen und dazu eine Route berechnen. Rechts sieht man eine Deutschlandkarte, wo die Route dann jeweils angezeigt wird. Außerdem hat man noch die Möglichkeit, das Ganze am Ende zu Exportieren.

6.2 Mängel

Einer der einzigen Mängel unseres Programms ist, dass wir kein dynamisches Filtern miteingebracht haben. Das heißt, dass wir die Suchergebnisse nicht dynamisch anzeigen lassen, sondern ganz normal. Dies waren in der Aufgabenstellung allerdings nur Brownie Punkte, also haben wir dies nun außen vorgelassen.

6.3 Fazit

Jesse: Am Anfang dachte ich, dass das meiste von Anhieb sehr leicht zu verstehen sein wird – allerdings verlief das Umrechnen von lat/lon in x/y nicht geplant. Demnach hat dies etwas länger gedauert, als erwartet – im Großen und Ganzen war ich allerdings erfolgreich. Ich konnte den Dijkstra-Algorithmus gut implementieren und habe ihn auch verstanden. Von der Gruppe her, habe ich schon oft mit Nele zusammengearbeitet und wir können unsere Schwächen / Stärken in verschiedenen Fächern gut ausgleichen.

Nele: Von meiner Seite aus gibt es eigentlich nicht viel zu sagen. Das Projekt verlief relativ normal und meine Zielsetzung habe ich erreicht – jedenfalls hab ich es geschafft, die GUI dieses Mal nicht in eine einzige Methode zu schreiben und bin daher sehr zufrieden. Mit Jesse habe ich bisher schon sehr oft in einer Gruppe gearbeitet und er ist einer der wenigen, mit denen man sehr gut zusammen arbeiten kann. Im Gegensatz zu anderen arbeitet er sehr viel - was einem zunächst Sorgen bereiten kann, aber wenn man vorher schon ausmacht wer in der Gruppe welche Aufgabe hat, läuft eigentlich alles wie geplant. Bei Nachfragen war er stets erreichbar und verständnisvoll.