Einführung:

Der Starship ist ein Roboter, der sich zur Zeit in der Testphase befindet. Er wurde konzipiert um ein Post- und Paketbote zu sein, um somit sämtliche Versandunternehmen zu entlasten. Mit seinen 21 Kilogramm Eigengewicht, fasst er eine Ladekapazität von etwa 15 Kilogramm, die er mit maximal 6 Kilometern pro Stunde, 5 Kilometer weit transportieren kann.

Ausgestattet ist er unter anderem mit neun Kameras, sechs Rädern und jeder Menge Technologie, um Gefahren zu erkennen und auszuweichen. Nichts desto trotz ist Starship in seiner Fortbewegung stark eingeschränkt, sodass er manche Straßen/Wege nicht befahren kann und vor allem auch nicht befahren darf. Darum haben wir es uns zur Aufgabe gemacht ich eine Routinglösung mithilfe des Graphhoppertools zu erstellen.

Vorgehensweise:

Download folgender Daten:

- GraphHopper Source Code (https://github.com/graphhopper/graphhopper)
- IDE Eclipse Umwicklungsumgebung (https://www.eclipse.org/downloads/)
- Download des Java Development Kit (JDK) Version 1.8 (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)

Somit haben wir das Grundgerüst für ein Routing, dass verschiedene Fortbewegungsmittel (zum Beispiel: Auto, zu Fuß oder Fahrrad) beinhaltet. Die erste Aufgabe bestand darin einen neuen FlagEncoder zu erstellen, der sämtliche Daten enthält, wo und wie unser Starship sich fortbewegen kann und darf.

Ein Flagencoder ist eine Klasse, in der alle Einstellungen gespeichert ist, auf welche Art und Weise sich das Fahrzeug fortbewegen darf und kann. Am wichtigsten hierbei sind die Einstellungen zur Geschwindigkeit und die Straßenerlaubnis.

Als Grundlage für den Starship-Flagencoder haben wir uns den FootFlagEncoder genommen, da sich die Fortbewegungsrichtlinien von Fußgängern sich am meisten ähneln mit denen unseres Roboters. Folgende Änderungen und Ergänzungen wurden vorgenommen:

Die Minimalgeschwindigkeit setzten wir auf 2km/h, die Durchschnittsgeschwindigkeit auf 4km/h und die Maximalgeschwindigkeit auf 7km/h. Grund dafür ist das hohe Menschenaufkommen in Berlin, so dass es dem Starship nicht möglich ist durchgehend auf der Maximalgeschwindigkeit zu fahren.

In den folgenden Screenshots unseres Codes sind die wichtigsten Änderungen und Ergänzungen zu entnehmen. Es wurde viel Wert darauf gelegt, wo der Starship lang fahren darf, da es dort viele Einschränkungen gibt. Auch Barrieren wurden mit eingefügt, um eine möglichst sichere Fahrt zu gewährleisten.

Absolute Barrieren:

```
setBlockByDefault(false);
absoluteBarriers.add("fence");
absoluteBarriers.add("cable_barrier");
absoluteBarriers.add("city_wall");
absoluteBarriers.add("ditch");
absoluteBarriers.add("fence");
absoluteBarriers.add("guard_rail");
absoluteBarriers.add("hedge");
absoluteBarriers.add("retaining_wall");
absoluteBarriers.add("wall");
absoluteBarriers.add("block");
absoluteBarriers.add("border_control");
absoluteBarriers.add("bump_gate");
absoluteBarriers.add("bus_trap");
absoluteBarriers.add("cattle_grid");
absoluteBarriers.add("debris");
absoluteBarriers.add("full-height_turnstile");
absoluteBarriers.add("horse_stile");
absoluteBarriers.add("jersey_barrier");
absoluteBarriers.add("kent_carriage_gap");
absoluteBarriers.add("kissing_gate");
absoluteBarriers.add("log");
absoluteBarriers.add("spikes");
absoluteBarriers.add("stile");
absoluteBarriers.add("sump_buster");
absoluteBarriers.add("yes");
```

Potentielle Barrieren:

```
potentialBarriers.add("gate");
potentialBarriers.add("cattle_grid");
potentialBarriers.add("handrail");
potentialBarriers.add("kerb");
potentialBarriers.add("tank_trap");
potentialBarriers.add("bollard");
potentialBarriers.add("chain");
potentialBarriers.add("cycle_barrier");
potentialBarriers.add("gate");
potentialBarriers.add("sliding_gate");
potentialBarriers.add("hampshire_gate");
potentialBarriers.add("lift_gate");
potentialBarriers.add("rope");
potentialBarriers.add("swing_gate");
```

Sichere Straßen/Wege:

```
safeHighwayTags.add("footway");
safeHighwayTags.add("path");
safeHighwayTags.add("pedestrian");
safeHighwayTags.add("sidewalk");
safeHighwayTags.add("crossing");
safeHighwayTags.add("traffic_signals");
```

Erlaubte Straßen/Wege

```
allowedHighwayTags.add("cycleway");
allowedHighwayTags.add("unclassified");
allowedHighwayTags.add("road");
allowedHighwayTags.add("passing_place");
allowedHighwayTags.add("residential");
allowedHighwayTags.add("living_street");
allowedHighwayTags.add("road");
allowedHighwayTags.add("bridleway");
allowedHighwayTags.add("elevator");
```

Vermeidete Straßen/Wege:

```
avoidHighwayTags.add("trunk");
avoidHighwayTags.add("trunk_link");
avoidHighwayTags.add("primary");
avoidHighwayTags.add("primary_link");
avoidHighwayTags.add("secondary");
avoidHighwayTags.add("secondary_link");
avoidHighwayTags.add("tertiary");
avoidHighwayTags.add("tertiary_link");
avoidHighwayTags.add("steps");
avoidHighwayTags.add("motorway");
avoidHighwayTags.add("motorway_link");
avoidHighwayTags.add("unclassified");
avoidHighwayTags.add("service");
avoidHighwayTags.add("bus_guideway");
avoidHighwayTags.add("escape");
avoidHighwayTags.add("proposed");
avoidHighwayTags.add("construction");
avoidHighwayTags.add("bus_stop");
```

Starten der GraphHopper Routing API

Um die GraphHopper Routing API auf einem Lokalen Server unter Windows 10 zu starten müssen folgende Schritte durchgeführt werden:

1. Öffnen der Kommandozeile
2. Dirigieren in das Verzeichnis, in dem die Dateien liegen
3. Folgenden Code eingeben:
    a. **Wichtig:** Aufgrund der Datengrößenbegrenzung auf 25 MB konnte nur die osm.pbf-Datei von Bremen auf Github hochgeladen werden. Demzufolge muss man in dem folgenden Code den Namen der osm.pbf Datei ändern oder sich über folgenden Link den Berlin Datensatz runterladen: https://download.geofabrik.de/europe/germany.html

```
D:\navigation\starshipneu>java -Dgraphhopper.datareader.file=berlin-latest.osm.pbf -jar graphhopper-web-0.12-SNAPSHOT.jar
 server config-example.yml
```

Konfigurationsdatei | OSM-Datei | Routingeinstellunge

Berlin:

java -Dgraphhopper.datareader.file=berlin-latest.osm.pbf -jar graphhopper-web-0.12-SNAPSHOT.jar server config-example.yml

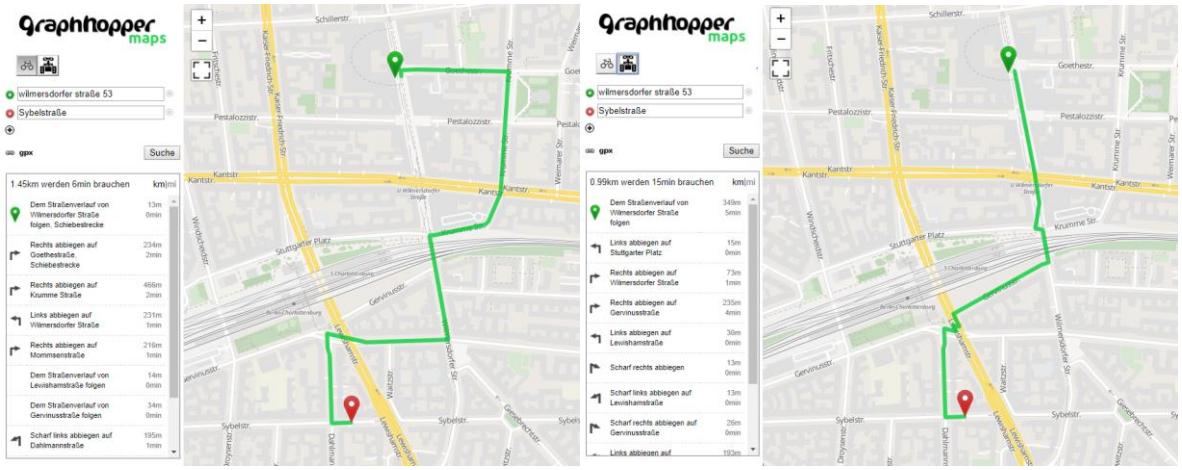
Bremen:

java -Dgraphhopper.datareader.file=bremen-latest.osm.pbf -jar graphhopper-web-0.12-SNAPSHOT.jar server config-example.yml

4. Localhost starten via: http://localhost:8989

Auf den folgenden Bildern wir das Routing von einer Postfiliale zu einem Wohnhaus in Berlin-Charlottenburg gezeigt. Im linken Bild ist das Fahrrad als Verkehrsmittel gewählt und im rechten der Starship Roboter.

Zu sehen ist, dass der Starship im Gegensatz zum Fahrrad eine Fußgängerzone entlangfahren darf, im Gegensatz zum Fahrrad, und somit einen kürzeren Weg hat. Der Weg beträgt beim Starship einen Kilometer und wird in 15 min abgefahren. Für kurze Strecken ist der Roboter also sehr gut geeignet.



Literaturverzeichnis:

https://ecomento.de/2015/11/10/dieser-elektrische-mini-roboter-soll-waren-ausliefern-video/