1. Aufgabenanalyse

# 1.1. Interpretation

Für die fiktive Reederei „MATSE-White-Star-Line“ soll der der Kurs eines Schiffes berechnet werden, dass sich durch ein Gebiet mit mehreren Strömungsfeldern von Punkt A nach Punkt E navigiert. Dabei soll das aktuelle Strömungsfeld so in den Kurs eingerechnet werden, dass sich diese beiden neutralisieren und effektiv der direkte Weg von A nach E mit einer Geschwindigkeit von 5 km/h genommen wird.

Dies wird auf ein2D-Koordinatensystem übertragen, wobei eine Längeneinheit einem Kilometer entspricht:

* Das Gebiet wir durch einen Bereich vom Ursprung (0;0) und einem weiteren Punkt definiert.
* Der effektive Kurs wird durch eine Strecke zwischen Punkt A und Punkt E dargestellt.
* Die Strömungsfelder werden durch einen Start- und einen End-Punkt, welche als die gegenüberliegenden Eckpunkte behandelt werden, und einem Strömungsvektor, der die Strömung in 1 km/h angibt, zusammengesetzt.

Außerdem können sich die Strömungsfelder überschneiden, hierbei hat immer das zuletzt angegeben wurde.

Der in der Aufgabenstellung beschriebene Lösungsansatz bestand darin, die direkte Route an ihren Schnittpunkten zu teilen und die Teilabschnitte, die nun nur durch eine konstante Strömung beeinflusst werden, zu berechnen.

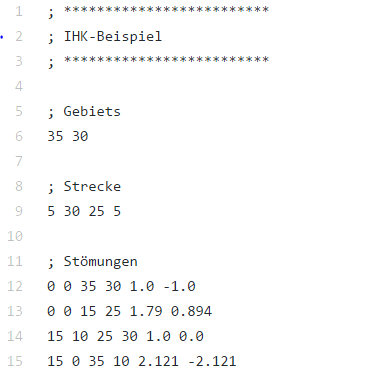
Ausgegeben sollte am Ende die Route, unterteilt in mehrere Teilabschnitte und für diese Teilabschnitte die effektive Länge, der zu fahrende Kurs mit der entsprechenden Geschwindigkeit, und die Zeit die für die effektive Länge mit der Geschwindigkeit gebraucht würde (dies wirkt wie ein sehr nichts aussagender Wert).

# 1.2. Eingabeformat

Die Eingabe besteht aus vier Teilen in dieser Reihenfolge:

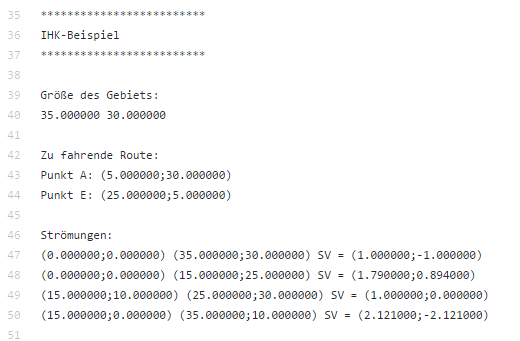
1. Eine Beschreibung komplett in Kommentaren verfasst. Eingeleitet durch einen Kommentar mit 25 ‚\*‘-Symbolen. Danach kommen beliebig viele Kommentarzeilen als Beschreibung, bis diese wieder mit 25 ‚\*‘-Symbolen beendet wird.
2. Das Gebiet wird durch zwei mit einem Leerzeichen getrennten Ganzzahlen angegeben, welcher den Punkt darstellt, der mit dem Ursprung das Gebiet abspannt.
3. Die Route wird durch vier mit einem Leerzeichen getrennten Ganzzahlen angegeben. Diese stehen für die vier Koordinaten-Werte der zwei Punkte, welche die Strecke bestimmen.
4. Die Strömungsfelder durch vier mit einem Leerzeichen getrennten Ganzzahlen und zwei Dezimalzahlen angegeben. Die Ganzzahlen stehen für die vier Koordinaten-Werte der zwei Punkte, welche das Feld eingrenzen und die beiden Dezimalzahlen stehen für die beiden Werte des Strömungsvektors

Kommentarzeilen werden durch ein Semikolon eingeleitet. Außerdem werden Leezeilen ignoriert.

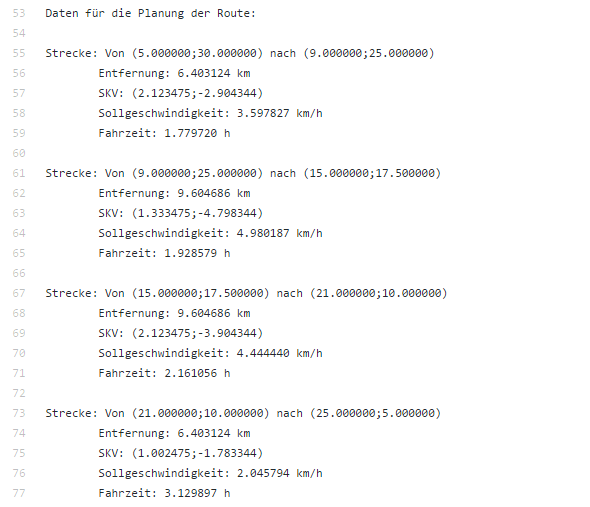


# 1.3. Ausgabeformat

Zunächst wird bei der Ausgabe die Eingabe wiederholt:



Dann werden nacheinander vom Start A zum Ziel E die Streckenabschnitte angegeben:



# 1.4. Fehlerfälle

Die auftretenden Fehler können in drei Fehlerarten aufgeteilt werden. Man unterscheidet zwischen technischen, syntaktischen und semantischen Fehlern.

Technische Fehler liegen vor, wenn die angegebene Datei nicht vorhanden ist, durch fehlende Zugriffsrechte nicht gelesen werden kann, oder die Ausgabedatei durch fehlende Schreibrechte nicht erstellt werden kann.

Syntaktische Fehler treten auf, wenn die Eingabedatei nicht der vorgegebenen Form entspricht, bei Semantische Fehler sind die Daten in der Eingabedatei fehlerhaft.

Durch die Analyse der Aufgabenstellung und des Eingabeformates ergeben sich folgende syntaktische Fehlerfälle:

* In der Beschreibung befinden sich nicht-Kommentar-Zeilen
* Die Beschreibung wird nicht gestartet und/oder beendet
* Es fehlen Werte zum einlesen

Durch die Analyse der Aufgabenstellung und des Eingabeformates ergibt sich nur ein semantischer Fehlerfall, dass die angegebene Route nicht im angegeben Gebiet liegt.

2. Verfahrensbeschreibung

# 2.1. Vorgehensweise

###### Einlesen

Zunächst wird versucht, alle Dateien im Input-Verzeichnis zu lesen, und aus allen ein valides Input zu parsen. Nun wird jedes fehlerfreie Input der Algorithmus gestartet.

###### Vorbereitung

Zuerst wird eine leere Liste an Routenabschnitten erzeugt, welche im weiteren Verlauf gefüllt wird. Dann wird ein *Stack* mit den Strömungsfeldern in der Reihenfolge gefüllt, in der sie eingegeben wurden, und eine Liste an Strecken mit zunähst nur der Route als Inhalt erstellt.

###### Algorithmus

Nun wird solange, wie sowohl noch Strömungsfelder als auch noch Strecken vorhanden sind, folgendes wiederholt:

Das nächste Strömungsfeld wird vom *Stack* genommen. Nun wird die Strecken-Liste durchlaufen. Für jede Strecke werden nun alle Schnittpunkte mit dem aktuellen Strömungsfeld berechnet. Hier gibt es drei Fälle

1. Keine Schnittpunkte, aber die Strecke ist komplett im aktuellen Feld. Dann wird ein Routenabschnitt mit der aktuellen Strecke und dem aktuellen Strömungsvektor der Lösungsliste hinzugefügt und die Strecke wird aus der Strecken-Liste gelöscht.
2. Ein Schnittpunkt, hier muss nochmal unterschieden werden zwischen
   1. Strecken-Start liegt im Feld, dann wird ein Routenabschnitt mit einer Strecke vom Start der aktuellen Strecke bis zum Schnittpunkt und dem aktuellen Strömungsvektor der Lösungsliste hinzugefügt und die aktuelle Strecke aus der Strecken-Liste gelöscht. Außerdem wird eine Strecke vom Schnittpunkt bis zum Ende der aktuellen Strecke nach dem Durchlauf der Strecken-Liste dieser hinzugefügt.
   2. Strecken-Ende liegt im Feld, dann wird ein Routenabschnitt mit einer Strecke vom Schnittpunkt bis zum Ende der aktuellen Strecke und dem aktuellen Strömungsvektor der Lösungsliste hinzugefügt und die aktuelle Strecke aus der Strecken-Liste gelöscht. Außerdem wird eine Strecke vom Start der aktuellen Strecke bis zum Schnittpunkt nach dem Durchlauf der Strecken-Liste dieser hinzugefügt.
3. Zwei Schnittpunkte, hier wird ein Routenabschnitt mit einer Strecke zwischen den Schnittpunkten und dem aktuellen Strömungsvektor der Lösungsliste hinzugefügt und die aktuelle Strecke aus der Strecken-Liste gelöscht. Außerdem werden sowohl eine Strecke vom Start der aktuellen Strecke bis zum entsprechend näheren Schnittpunkt als auch eine vom anderen Schnittpunkt bis zum Ende der aktuellen Strecke nach dem Durchlauf der Strecken-Liste dieser hinzugefügt.

###### Restdaten

Sollten nach dem Algorithmus noch Strecken in der Strecken-Liste sein, wird für jede von ihnen ein Routenabschnitt mit der Strecke und dem Strömungsvektor *(0;0)* der Lösungsliste hinzugefügt.

###### Sortierung

Zuletzt wird die Ergebnisliste noch vom Start bis zum Ende sortiert.

# 2.2. Gesamtsystem

###### 2.2.1. Main-Funktion

Die *main*-Funktion ruft verarbeitet die Daten, die ihr von den Klassen zugetragen wird. Sie ruft das Auslesen der *Input*-Klasse auf, verarbeitet die Daten mit den Methoden der *Dataholder*-Klassen und gibt schlussendlich die Ergebnisse an den *Output* weiter.

###### 2.2.2. Input

Die *Input*-Klasse kann statisch den Input-Ordner auslesen und damit Objekte von sich selbst erzeugen.

###### 2.2.2. Dataholder

Die *Dataholder*-Klassen stellen die geometrischen Objekte für die Berechnungen dar und können mit sich selbst und miteinander rechnen.

###### 2.2.3. Output

Die *Output*-Klasse kann die Ergebnisse in die *output.txt*-Datei schreiben.

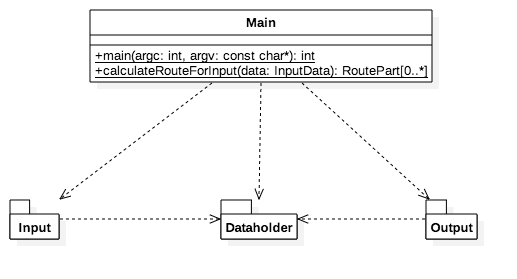
# 2.2. Datenfluss

>Sequenz-Diagramm

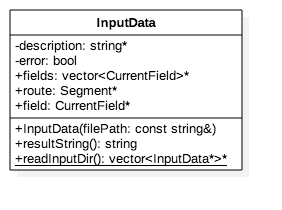
3. Programmbeschreibung

# 3.1. Pakete

Das Programm ist in drei Module aufgeteilt, welche dem EVA-Model entsprechen. Für jedes Modul wurde ein entsprechender *namespace* deklariert.

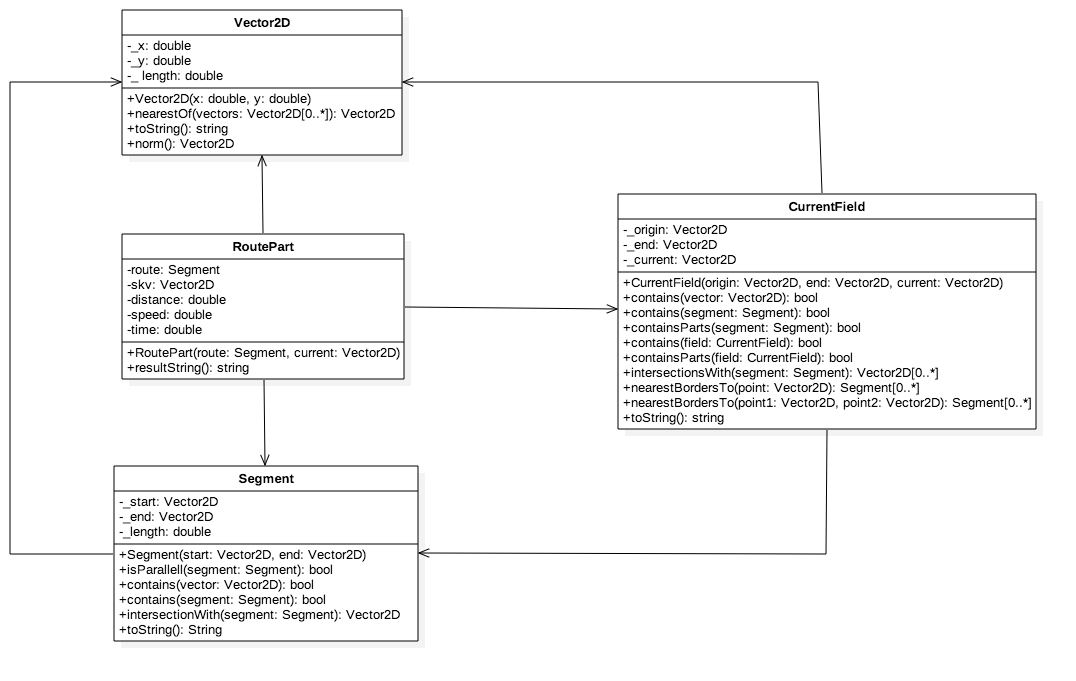


###### 3.1.1. Input



**InputData:** Kapselung des Lesens der Eingabedateien.

###### 3.1.2. Dataholder



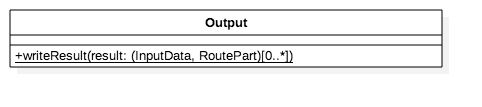
**Vector2D:** Abstrahierung eines 2D-Vectors.

**Segment:** Abstrahierung einer Strecke im 2-Dimensionalen Raum.

**CurrentField:** Abstrahierung der Strömungsfelder.

**RoutePart:** Kapselung der Berechnung der Ergebnisse.

###### 3.1.2. Output



**Output:** Kapselung des Schreibens der Ergebnisse.

# 3.2. Präzisierung

###### 3.2.1. Segment.intersectionsWith(Segement segment)

# ../Downloads/intersectionWith.png

In der Methode intersectionsWith(Segement segment) der Segment-Klasse wird der mögliche Schnittpunkt zwischen der übergebenen Strecke und der, auf dem die Methode aufgerufen wird, ermittelt.

Dafür wird zunächst geprüft, ob die Strecken parallel sind oder sich nur berühren. Falls eins der Fall ist, wird sofort abgebrochen.

Zur Berechnung werden die Strecken wie geraden behandelt und die Cramerische Regel angewendet. Diese hat im 2-dimensionalen für 2 Geraden zwischen je 2 Punkten folgende geschlossene Darstellung:

Sollte währenddessen im Nenner 0 sich ergeben, wird sofort abgebrochen.

Wenn dann ein Schnittpunkt gefunden wurde, wird noch überprüft, ob dieser auch auf beiden Strecken liegt, da wir vorher mit Geraden gearbeitet haben. Nur in diesem Fall wird der Punkt zurückgegeben.

###### 3.2.1. CurrentField.intersectionsWith(Segement segment)



In der Methode intersectionsWith(Segement segment) der CurrentField-Klasse werden die möglichen Schnittpunkte zwischen der übergebenen Strecke und dem aufgerufenen Feld ermittelt.

Zunächst wir geprüft, ob die Strecke im Feld enthalten ist. In diesem Fall wird eine leere Liste zurückgegeben, da keine Schnittpunkte existieren können.

Nun werden für alle Punkte die außerhalb des Felds liegen, die jeweils ein bis zwei nächsten Kanten des Feldes genommen und mit diesen alle möglichen Schnittpunkte ermittelt.

Fall dies nicht die maximale Anzahl an zwei Schnittpunkten überschreitet, werden diese zurückgegeben.