

Landkarten

Dokumentation der großen Programmierarbeit



access

powered by technology

Autor: Milan Kanz
Status: MATSE Auszubildender

Inhaltsverzeichnis

1	Aufgabenanalyse.....	3
1.1	Problembeschreibung.....	3
2	Verfahrensbeschreibung.....	5
3	Programmbeschreibung.....	7
3.1	Aufbau nach EVA-Prinzip	7
3.2	Formale Beschreibung des Algorithmus.....	10
4	Benutzeranleitung.....	14
4.1	Ordner- und Dateistruktur	14
4.2	Start des Programms	14
5	Entwicklerdokumentation	15
5.1	Klassen und Schnittstellen.....	15
5.2	Nutzung.....	15
6	Testfälle.....	16
6.1	Normalfälle.....	16
6.2	Sonderfälle.....	16
7	Beschreibung der Entwicklungsumgebung.....	17
7.1	Rechner.....	17
7.2	Entwicklungsumgebung	17
7.3	Compiler.....	17
7.4	Dokumentation	17
8	Abweichungen vom Konzept	18
8.1	Abweichungen und Ergänzungen des Algorithmus	18
8.2	Abweichungen in der Klassenstruktur.....	18
9	Zusammenfassung und Ausblick.....	19
10	Abbildungsverzeichnis	20

1 Aufgabenanalyse

1.1 Problembeschreibung

In der hier vorliegenden Aufgabe soll ein Algorithmus zur Berechnung Staatspezifischer Kennwerte unter Berücksichtigung einzulesender Lage- und Nachbarbeziehungen entwickelt werden. Die resultierenden Ergebnisse sollen in einer Ausgabedatei zur Darstellung zu Verfügung gestellt werden. Die Qualität der Karte misst sich an der möglichst guten Lagebeziehungen unter dem Gesichtspunkt dass Überschneidungen zu vermeiden bzw. verringern sind und Nachbarstaaten einen möglichst geringen Abstand zueinander haben sollen. Die Darstellung der Staaten soll lediglich Schemenhaft passieren, weshalb sich Kreisförmig aufgrund geeigneter mathematischer Recheneigenschaften anbietet. Der Kennwert entspricht hierbei der Kreisfläche, worüber der Radius leicht zu berechnen ist.

Die Koordinatengrößen sind beliebig änderbar, wichtig ist dass die Relationen erhalten bleiben und nur leicht angepasst werden. Diese Anpassung findet über Anziehungs- und Abstoßungskräfte in einem iterativen Prozess statt. Die Kräfte errechnen sich aus dem Abstand zweier benachbarter Kreise/Staaten und werden für jeden Nachbarn berechnet.

Der Richtungsvektor errechnet sich aus den Kreismittelpunkten. Abstoßungs- und Anziehungskraft unterscheiden sich in der Polarität und Stärke, bezeichnen aber letztlich den gleichen Zusammenhang zweier Nachbarn. Überschneiden sich die Kreise zweier Nachbarn, also bei einem negativen Abstand, handelt es sich um Abstoßungskraft und bei Abständen größer Null um Anziehungskraft. Diese Kräfte können als einfacher Vektor modelliert und gespeichert werden.

Sollten mehrere Nachbarstaaten auf den gleichen Koordinaten sein ist eine sinnvolle Abstands- bzw. Richtungsberechnung nicht möglich und die Eingabe wird als Fehlerhaft bewertet. Wird ein Land ohne Nachbarn in der Eingabedatei gelistet gilt dies als Fehlerhaft da keine Kräfteberechnung möglich ist. Das Land wird bei der weiteren Berechnung nicht berücksichtigt.

Es kann unter Umständen dazu kommen dass ein Land sich (durch Verschiebung oder Eingabe) mit einem nicht benachbarten Land überschneidet. Dies wird Programmatisch nicht erkannt da die Abstands Berechnungen nur für Nachbarstaaten durchgeführt werden. Aufgrund der Seltenheit bzw. Unwahrscheinlichkeit wird dies als theoretische Möglichkeit hingenommen ohne überprüft bzw. behandelt zu werden.

Gibt es nur ein einziges Land wird das Programm abgebrochen da keine Nachbarschaftsberechnung möglich ist.

Eingabedatei:

Die Eingabedatei beginnt mit dem Namen des Kennwertes der später bei der Ausgabe angegeben werden soll. Es ist nicht angegeben ob es sich hierbei auch um eine Mehrzeilige Bezeichnung handeln kann, daher wird davon ausgegangen dass alle Zeilen bis zur ersten Kommentarzeile Teil dieser Bezeichnung sind. Als Kommentarzeile gelten alle Zeilen die mit dem Zeichen „#“ beginnen. Diese Zeilen werden nicht eingelesen und dienen einzig der

Unterteilung in die verschiedenen Datenabschnitte. Ein Kommentar kann auch am Ende einer Datenzeile stehen, es wird wieder durch „#“ gekennzeichnet, diese Kommentare gelten nicht als Kommentarzeile. Es werden automatisch alle Datenzeilen auf dieses Symbol überprüft und ggf. werden die Kommentare ausgefiltert.

Im Anschluss werden Zeilenweise die Informationen der Staaten in jeweils einem eigenen *Staat*-Objekt abgespeichert. Die Reihenfolge der Daten ist wie folgt: Kennzeichen des Landes, Kennwert, Längengrad und Breitengrad, die Werte sind mit Leerzeichen voneinander getrennt. Wird eine zweite Kommentarzeile gefunden gilt das Einlesen der Staaten als beendet. Nach der Kommentarzeile werden die Nachbarschaftsbeziehungen eingelesen. Eine Zeile beginnt mit dem Kennzeichen eines Staates und einer Liste der Nachbarstaaten im Anschluss an das „:“-Zeichen. Die Liste der Nachbarstaaten ist mit Leerzeichen voneinander getrennt. Bei der Zuordnung der Nachbarschaftsbeziehungen ist zu beachten dass die Angabe bidirektional ist, z.B. muss bei der Zuordnung dass „NL“ Nachbar von „D“ ist, ebenfalls eingetragen werden dass „D“ Nachbar von „NL“ ist. In der Abbildung 1 ist ein Beispiel einer Eingabedatei zu sehen.

```
Fläche der Staaten
# Staat Fläche Längengrad Breitengrad
D      357    10.0   51.3
NL     42      5.3    52.2
B      33      4.8    50.7
L      3       6.1    49.8
F      544    2.8    47.4
CH     41      8.2    46.9
A      84      14.2   47.6
CZ     79      15.3   49.8
PL     313    18.9   52.2
DK     43      9.6    56.0
# Nachbarschaften
D: NL B L F CH A CZ PL DK
NL: B
B: L F
L: F
F: CH
CH: A
A: CZ
CZ: PL
```

Abbildung 1: Beispiel einer Eingabedatei

Es kann davon ausgegangen werden dass die Eingabedatei den syntaktischen Vorgaben entspricht, daher muss lediglich bei der Konvertierung in die gewählten Datentypen eine Überprüfung und Fehlerbehandlung in der Input-Klasse berücksichtigt werden.

Ausgabedatei:

Die Ausgabedatei entspricht einem vorgegebenen Gerüst in das bestimmte Werte eingetragen werden.

```

reset
set xrange [<xmin>:<xmax>]
set yrange [<ymin>:<ymax>]
set size ratio 1.0
set title "<Name des Kennwertes>, Iteration: <n>""
unset xtics
unset ytics
$data << EOD
<Liste aus <xpos> < ypos> < radius> < kennzeichen> < id>>
EOD
plot \
'$data' using 1:2:3:5 with circles lc var notitle, \
'$data' using 1:2:4:5 with labels font "arial, 9" tc variable notitle

```

Abbildung 2: Gerüst der Ausgabedatei

Die in „< >“ eingeklammerten Werte sollen durch die errechneten Werte des Programms ersetzt werden.

<xmin>, <xmax>, <ymin>, <ymax> - Diese Werte definieren den Darstellungsbereich, hierbei soll $x_{\text{max}} - x_{\text{min}} = y_{\text{max}} - y_{\text{min}}$ gelten, also ein quadratischer Bereich definiert werden.

<Name des Kennwertes> - Die Bezeichnung die anfangs aus der Eingabedatei ausgelesen wird.

<n> - Die Anzahl der Iterationen die das Programm ausgeführt hat um zu dem aktuellen Ergebnis zu kommen.

<Liste aus ...> - Hier wird für jeden Staat eine Zeile mit den nachfolgenden Werten ausgegeben:

<xpos> - Die X-Koordinate des Kreismittelpunkts.

<ypos> - Die Y-Koordinate des Kreismittelpunkts.

<radius> - Der Radius des Kreises.

<kennzeichen> - Das Kennzeichen des Staats.

<id> - Eine fortlaufende ID zur Farbgebung in gnuplot. Die ID wird erst bei der Ausgabe erstellt, da das Programm vorher keine Verwendung dafür hat.

2 Verfahrensbeschreibung

Zur Ermittlung der Darstellung realer Nachbarschafts- und Lagebeziehungen nach Kennwert wird ein iterativer Vorgang angewandt um die Ergebnisse stetig zu verbessern. Vor der ersten Iteration wird die Qualität der übergebenen Daten, bzw. die Kartenausgangslage überprüft. Das hierbei gewählte Prüfkriterium ist dass die Summe aller Abstände, also die Summe aller Anziehungskräfte, größer als die Summe aller Überschneidung, also die Summe aller Abstoßungskräfte, sein soll (Details der Kräfteberechnung werden später erläutert). Kurzgesagt auf der Karte sollen global höhere Anziehungskräfte als Abstoßungskräfte herrschen. Ist dies nicht der Fall wird der Kennwert für alle Staaten auf die Hälfte skaliert, die Koordinaten der Staaten und dementsprechend sowohl Nachbarschafts- als auch Lagebeziehungen bleiben hierbei unverändert. Dadurch dass die Skalierung auf alle Staaten angewandt

wird, bleibt die Größe eines Staates relativ zu der Größe der anderen Staaten unverändert. Wie in der Aufgabenstellung ausdrücklich erlaubt wird also nur die numerische Größenordnung nach Bedarf angepasst. Diese Skalierung wird solange durchgeführt bis das genannte Prüfkriterium erfüllt ist.

Nach dieser Vorbereitung beginnt die Iterationsschleife. Als Abbruchbedingung gibt es zwei verschiedene Kriterien. Wie für das obige Prüfkriterium werden wieder die Summen der Kräfte voneinander subtrahiert, allerdings wird diesmal auf ein ausgeglichenes Verhältnis kontrolliert. Liegt der Betrag der Subtraktion unter einem bestimmten Toleranz-, bzw. Abweichungswert wird die Karte als qualitativ gut angesehen und das Programm beendet die Schleife. Sollte dieser Toleranzwert nicht erreicht werden schließt die Schleife ab wenn eine maximale Anzahl von Iterationen erreicht wurde. Die beiden Abbruchwerte werden Standardmäßig mit einem Default Wert initialisiert, alternativ können die Werte beim Programmaufruf ebenfalls vom Nutzer als Zusatzparameter angegeben werden.

Und nun zur Beschreibung der eigentlichen Iteration. Zunächst wird über die Liste alle Staaten iteriert und für jeden Staat die interne Funktion zur Kräfteberechnung aufgerufen. Die Funktion setzt zunächst die Anziehungs- und Abstoßungskräfte aus der Vorbereitung und eventuellen Voriterationen zurück. Es wird für jeden Nachbarstaat der relative Richtungsvektor berechnet und normiert abgespeichert. Die Länge des Richtungsvektors entspricht dem Abstand der Mittelpunkte des Staates und jeweiligem Nachbar. Mithilfe des Mittelpunktabstands und der Radien der beiden Staaten wird der Kreisrandabstand wie folgt berechnet:

$$\text{Kreisrandabstand} = \text{Mittelpunktabstand} - \text{Radius des Ausgangsstaats} - \text{Radius des Nachbarstaats}$$

Der Kreisrandabstand bestimmt sowohl die Art der Kraft (Anziehung oder Abstoßung), als auch die Stärke. Ist das Ergebnis positiv ist ein realer Abstand vorhanden und es handelt sich um Anziehungskraft, ist das Ergebnis negativ überschneiden sich die Kreise und es handelt sich um Abstoßungskraft. Dieser Abstand wird als Stärke des Vektors abgespeichert und beim Anfordern der Koordinaten verrechnet, wodurch eine automatische Umkehrung des Richtungsvektors stattfindet, sollte es ein Vektor mit Abstoßungskraft sein. Für die Qualitätsüberprüfung werden die Vektoren je nach Art der Kraft separat abgespeichert. Der Grenzfall dass der Abstand gleich Null ist muss nicht gesondert behandelt werden da diese Vektoren ohnehin keinen Einfluss auf das Ergebnis haben, Nullvektoren werden daher zusammen mit Anziehungsvektoren abgespeichert.

Nachdem die Kräfte für alle Staaten berechnet wurde müssen diese Kräfte jeweils auf alle Staaten angewendet werden. Hierzu wird nochmals über alle Staaten iteriert und jeweils die interne Funktion zum Anwenden der berechneten Kräfte aufgerufen. Die Funktion summiert alle, mit der entsprechenden Stärke und dadurch Polarität verrechneten, X- und Y-Koordinaten der Abstoßungs- und Anziehungsvektoren. Hierbei ist es wichtig zu beachten dass diese Kräfte nur anteilig und in Abhängigkeit der Anzahl der Nachbarn des jeweiligen Staats angewandt werden. Werden die Kräfte lediglich halbiert wird ein Staat mit mehr als einem Nachbar ggf. deutlich höheren Kräften ausgesetzt und die ursprünglichen Lagebeziehungen gehen verloren. Die Summe aller X- und Y-Verschiebungen werden letztlich jeweils auf die Koordinaten des aktuellen Mittelpunkts addiert und es ergibt sich die neue Lage.

Anschließend wird der Darstellungsbereich neu ermittelt und die Iteration ist abgeschlossen.

3 Programmbeschreibung

3.1 Aufbau nach EVA-Prinzip

Der Grundaufbau des Programms verfolgt das Eingabe, Verarbeitung, Ausgabe Prinzip (kurz: EVA). Die Module sind nach der jeweiligen Funktionalität getrennt und werden in der Hauptklasse einzeln durchlaufen.

Zunächst wird das beim Programmaufruf benötigte Pfadargument an die Eingabe-Klasse zum Einlesen übergeben. Der Reader speichert eine Liste der Staaten und den Namen des Kennwertes ab. Über get-Methoden können diese Daten in der Main-Klasse abgerufen und nach Bedarf an die anderen Module weitergegeben werden.

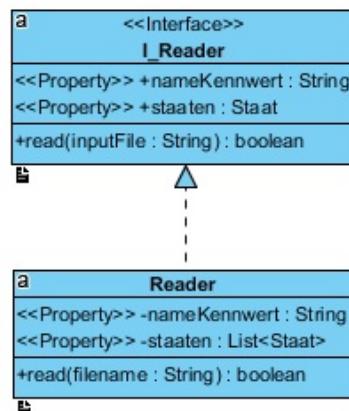


Abbildung 3: Struktogramm des Eingabe-Moduls

Die Informationen der Staaten werden beim Einlesen im Eingabemodul in Objekte der Klasse **Staat** abgespeichert. **Staat** bietet dem Hauptalgorithmus verschiedene Methoden zur Lösung an, darunter vor allem die Kräfteberechnung, Anwendung der Kräfte, Skalierung des Kennwertes bei der Iterationsvorbereitung und verschiedene Getter.

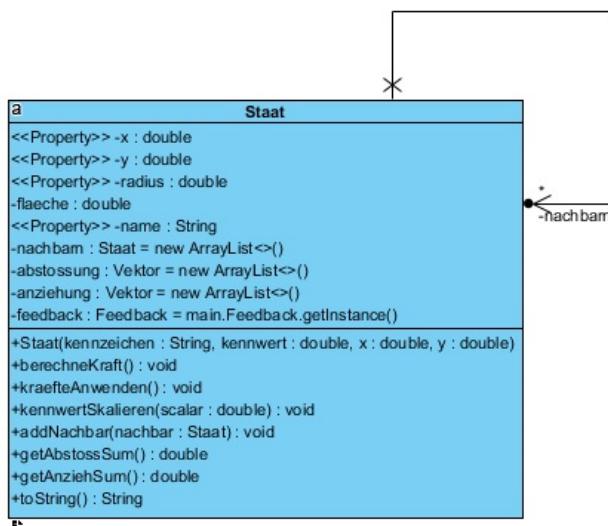


Abbildung 4: Struktogramm der Staat-Klasse

Die Kräfte werden in der Vektor-Klasse gespeichert.

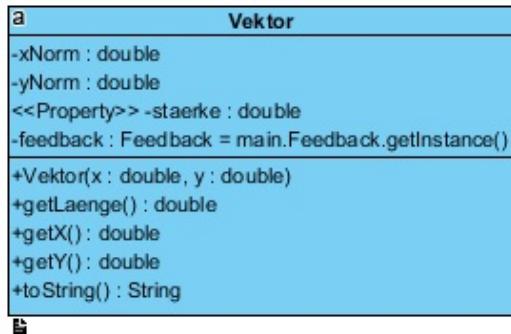


Abbildung 5: Struktogramm der Vektor-Klasse

Der Hauptalgorithmus *Laenderberechnung* wird mit der Liste der Staaten instanziert und über Aufruf einer Methode wird anschließend der Berechnungsalgorithmus ausgeführt. Die Staat-Objekte aus der Liste werden iterativ modifiziert und können anschließend wieder über einen Getter abgerufen werden. Außerdem werden noch weitere Werte wie der Darstellungsbereich in Form der minimalen und maximalen X und Y-Koordinaten, sowie die Anzahl der durchlaufenen Iterationen gesetzt und zur Verfügung gestellt.

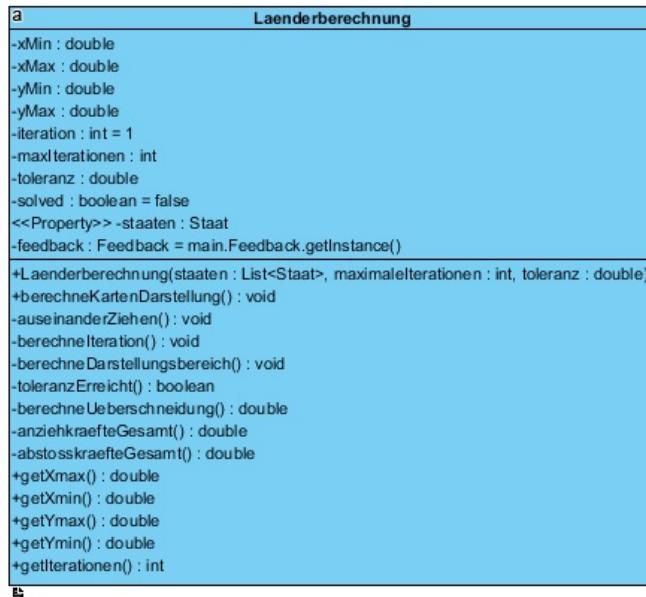


Abbildung 6: Struktogramm der Laenderberechnung-Klasse

Nach Durchführung der Berechnung wird in *Main* ein Objekt der Output-Klasse *Writer* erstellt und das Verarbeitungs-Objekt übergeben um auf die notwendigen Berechnungsergebnisse und Bedingungen Zugriff zu haben. Im Ausgabe-Modul wird mithilfe dieser Daten und des aus der Aufgabenstellung bekannten Gerüsts die Ausgabedatei erstellt.

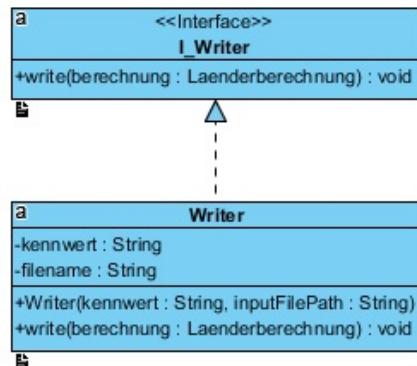


Abbildung 7: Struktogramm des Ausgabe-Moduls

Zusätzlich gibt es noch eine *Feedback*-Klasse die dem Nutzer über die Kommandozeilenschnittstelle Informationen wie Hilfestellung, Fehlermeldungen und Ergebnis-Informationen zurückgibt.

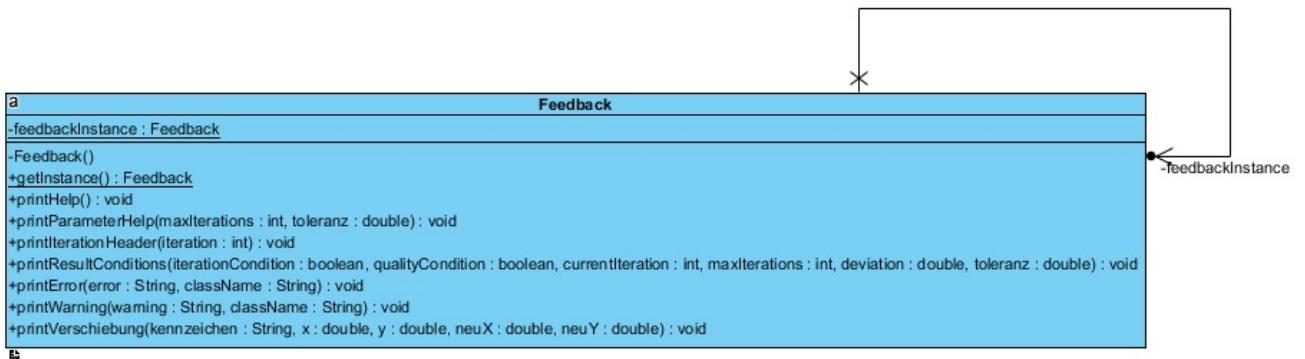


Abbildung 8: Struktogramm der Feedback-Klasse

Die Gesamtstruktur des Programmes sieht wie folgt aus:

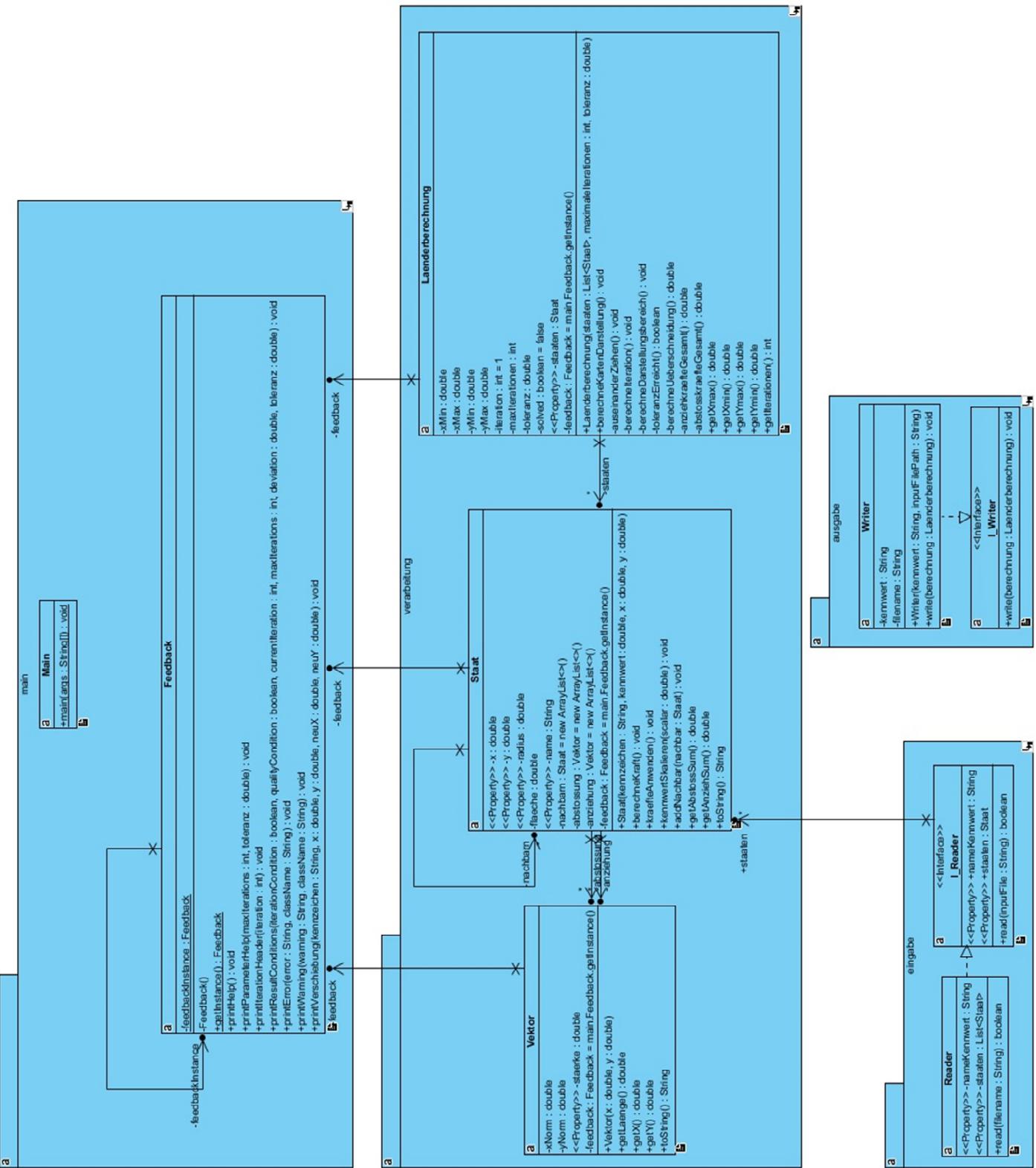


Abbildung 9: Klassenstruktur im Überblick

3.2 Formale Beschreibung des Algorithmus

Im Folgenden werden die Wesentlichen Methoden anhand von Nassi-Shneiderman-Diagrammen vorgestellt.

Die Methode welche die Iterationen verwaltet, mit Vorbereitung auf erste Iteration.

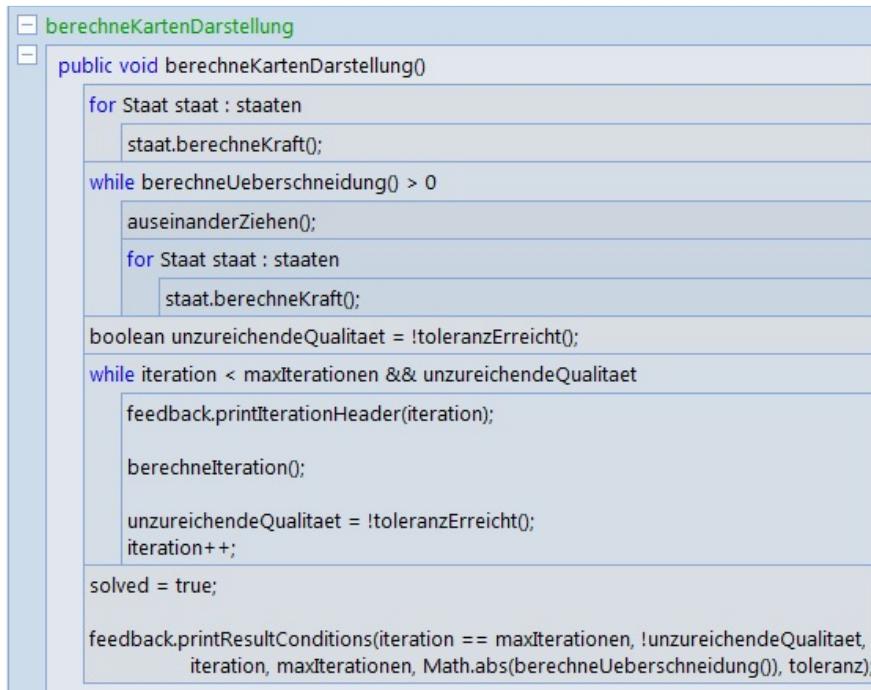


Abbildung 10: NS-Diagramm von berechneKartenDarstellung

Die Methode zum Berechnen der Kräfte.

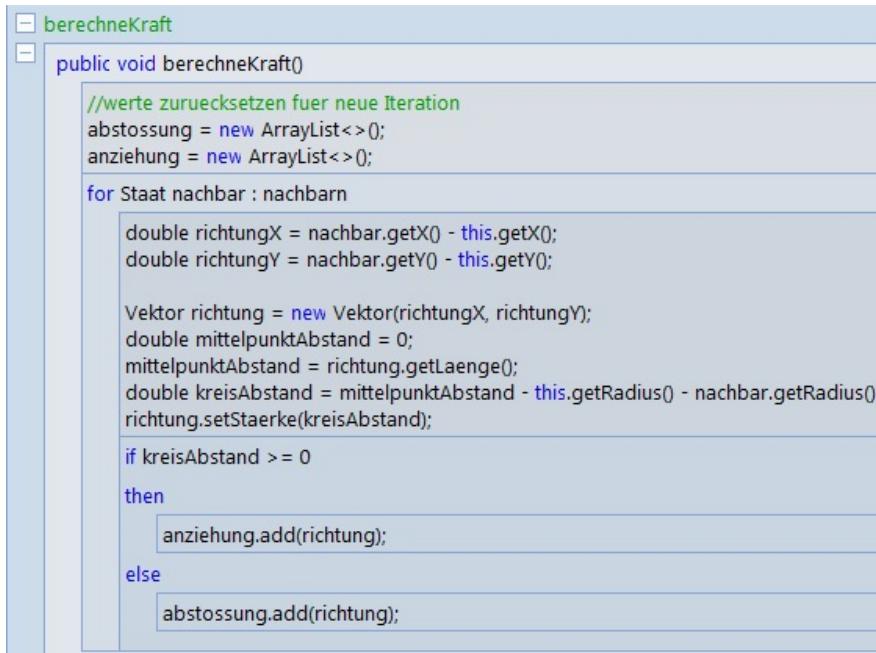


Abbildung 11: NS-Diagramm von berechneKraft

Die Methode der einzelnen Iteration.

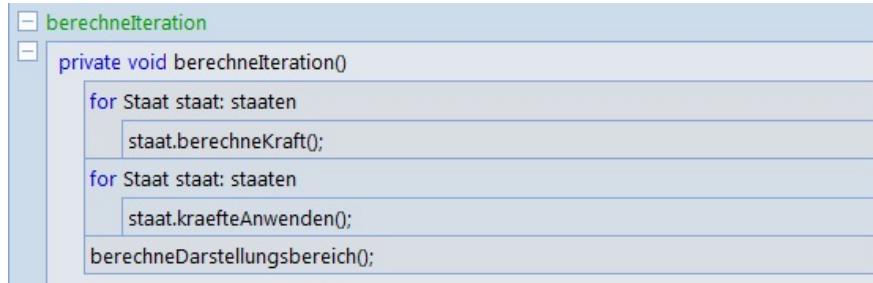


Abbildung 12: NS-Diagramm von berechneIteration

Die Methode zum Anwenden der Kräfte.

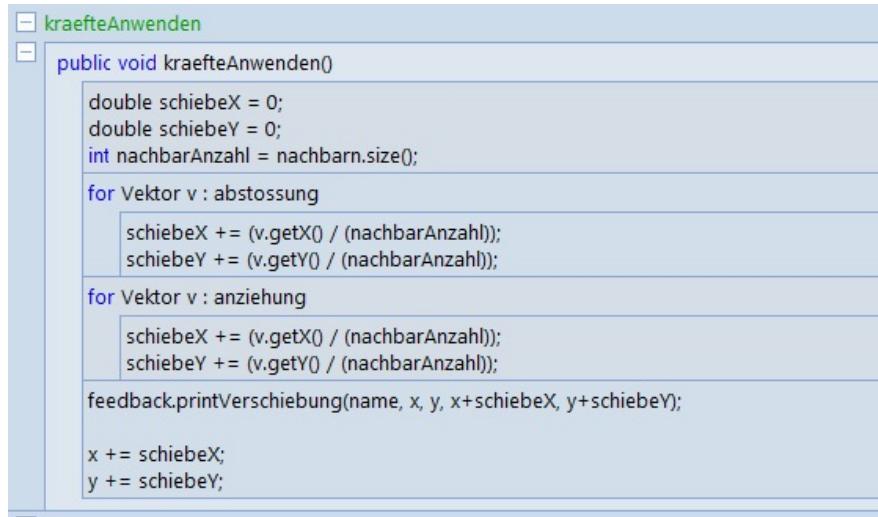


Abbildung 13: NS-Diagramm von kraefteAnwenden

Die Methode zur Berechnung des Darstellungsbereichs jeder Iteration.



Abbildung 14: NS-Diagramm von `berechneDarstellungsbereich`

4 Benutzeranleitung

Im Folgenden wird die Ausführung des Programms als jar-Datei und mithilfe eines Batch-Skripts beschrieben.

4.1 Ordner-und Dateistruktur

Im Ordner „executable“ der Abgabe befindet sich das kompilierte Programm als ausführbare jar-Datei. In dem Verzeichnis befindet sich außerdem ein Inputordner mit den beschriebenen Testfällen, sowie ein Outputordner in dem die Ergebnisdateien abgelegt werden. Die Output-Dateien werden zur Differenzierung mit einem Zeitstempel markiert. Es wird außerdem ein Batch-Skript zur Verfügung gestellt welches es ermöglicht das Programm für alle Dateien in einem Ordner auszuführen.

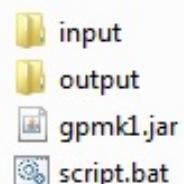


Abbildung 15: Ordnerstruktur

4.2 Start des Programms

Die Ausführung des Programms findet über die Kommandozeilenschnittstelle statt. Ein einfaches Ausführen des Programms mit einer Input-Datei kann z.B. wie folgt aussehen:

```
java -jar gpmk1.jar .\input\bsp1.in
```

Das Programm benötigt einen Pfad zu einer Eingabedatei, wird dieser nicht angegeben wird eine Hilfestellung zur Ausführung des Programms ausgegeben und das Programm bricht ab. Der Aufruf kann noch um die Abbruchbedingungen des Programms erweitert werden:

```
Java -jar gpmk1.jar <Dateipfad> <maximale Iterationen> <Toleranz>
```

Die maximale Anzahl von Iterationen muss als Ganzzahl angegeben werden, die Toleranz kann auch eine Gleitkommazahl sein. Diese zusätzlichen Parameter werden nicht benötigt und das Programm benutzt Default-Werte wenn sonst nichts angegeben ist.

Wie bereits angesprochen kann das Programm auf alle Dateien in einem Ordner angewendet werden mithilfe eines Batch-Skripts, die Ausgabe ist folgendermaßen aufgebaut:

```
script.bat gpmk1.jar -d <Ordnerpfad> <maximale Iterationen> <Toleranz>
```

Die hinteren beiden Parameter sind wie bei der normalen Ausführung Optional und können weggelassen werden.

5 Entwicklerdokumentation

Im Ordner „doc“ ist eine ausführliche Dokumentation der Klassen und Methoden als javadoc in HTML-Form abgelegt. Hier daher nur eine kurze Auflistung und Beschreibung der Klassen und Schnittstellen.

5.1 Klassen und Schnittstellen

- Interface I_Reader
Schnittstelle zum Einlesen der Eingabedatei
Klasse Reader
- Interface I_Writer
Schnittstelle für die Ausgabe der Plotdaten
Klasse Writer
- Klasse Staat
Speicherung der Staat-Informationen, beinhaltet Methoden zur Kräfteberechnung und Anwendung
- Klasse Vektor
Speicherung der normierten X- und Y-Koordinaten und der Stärke entsprechend des Kreisabstands
- Klasse Feedback
Rückgabe von Informationen und Fehlermeldungen über die Kommandozeilenschnittstelle
- Klasse Laenderberechnung
Iterative Berechnung des Haupalgorithmus
- Klasse Main
Programmausführung

5.2 Nutzung

Die Schnittstellen zur Ein- und Ausgabe dienen der einfacheren Erweiterung und Verwendung für andere Entwickler oder Rahmenbedingungen. Wenn die Daten in einem anderen Format abgelegt sind kann eine zusätzliche Klasse entwickelt werden welche die I_Reader Schnittstelle implementiert und entsprechend des abweichenden Formats die Daten einliest.

Ebenso kann die Ausgabe des Programms auf ein anderes plot Format zugeschnitten werden, indem die I_Writer Schnittstelle von einer neuen Klasse implementiert wird die eine angepasste Ausgabedatei erstellt.

6 Testfälle

Im Folgenden werden einige Testfälle vorgestellt und der Umgang des Programms bei Fehlern. Die Eingabedateien sind im Input-Ordner enthalten.

6.1 Normalfälle

Normalfälle sind Eingabedateien die ohne Fehlermeldung ausführen und ein sinnvoll nutzbares Ergebnis liefern. Hier sind die Beispiele aus der Aufgabenstellung mit enthalten. Alle Tests wurden mit einer maximalen Iterationszahl von 200 und einer Toleranz von 0.01 ausgeführt.

Testfall bsp1 bis bsp3

Die Beispiele aus der Klausur werden Problemlos und wie erwartet ausgeführt.

Testfall bsp4

Dieses Beispiel wurde zur simplifizierten Testung der Grundfunktionen modelliert. Es wird ein qualitativ gutes Ergebnis ausgegeben.

6.2 Sonderfälle

Sonderfälle sind Eingabedateien mit unüblichen Eigenschaften die mit oder ohne Fehlermeldung ausgeführt werden. Die Ergebnisse variieren stark, können nutzbar sein.

Testfall bsp5

Der Kennwert ist für alle Staaten identisch. Das Ergebnis für Nachbarstaaten ist von annehmbarer Qualität, allerdings kommt es zu großen Überschneidungen von nicht-Nachbarn.

Testfall bsp6

Die Nachbarn NL und B sind in der Eingabedatei auf der gleichen Position. Das Programm gibt in der ersten Iteration Fehlermeldungen aus da eine Abstandsbestimmung in diesem Fall nicht möglich ist. Ein Richtungsvektor mit Länge kann nicht erstellt werden und die Nachbarschaftsbeziehung wird für diese Iteration übersprungen. Aufgrund wirkender Kräfte von anderen Nachbarn verschieben sich die Positionen und die Kräfteberechnung zwischen NL und B wird ab der 2ten Iteration normal fortgesetzt. Es wird ein qualitativ gutes Ergebnis geliefert.

Testfall bsp7

Die nicht-Nachbarn B und A sind in der Eingabedatei auf der gleichen Position. Das Programm führt ohne Fehlermeldung aus, allerdings sind im Ergebnis viele Überschneidungen und die reale Lagebeziehung geht verloren.

Testfall bsp8

Ein vereinfachtes Beispiel zur Veranschaulichung wie die aktuelle verwendete Abbruchbedingung ausgetrickst werden kann. Dadurch dass der Algorithmus nur die globalen Kräfte überprüft kann eine Überschneidung von zwei

Staaten durch einen gleich großen Abstand von zwei anderen Staaten ausgeglichen werden in Fällen in denen eine bessere Lösung ohne komplizierte Berechnungen möglich wäre.

7 Beschreibung der Entwicklungsumgebung

7.1 Rechner

Die Entwicklung der Software und Erstellung der Dokumentation wurde auf einem Laptop mit folgenden Systemeigenschaften erarbeitet:

Prozessor: Intel® Core™ i5-6200U CPU @ 2.30GHz

Arbeitsspeicher: 8 GB

Systemtyp: 64 Bit-Betriebssystem

Betriebssystem: Windows 7 Enterprise, SP1

7.2 Entwicklungsumgebung

Die Software wurde in der Java IDE (Integrierte Entwicklungsumgebung) NetBeans entwickelt. Es wurde die NetBeans version 8.2 verwendet. Das Programm kann unter <https://netbeans.org/> heruntergeladen werden.

7.3 Compiler

Der Quellcode wurde mit dem JDK 1.8.0_101 innerhalb von NetBeans kompiliert. Es ist zu beachten dass die commons.io-2.6 Bibliothek benutzt wird und diese mit in die jar-Datei gebaut werden. Downloadlink: https://commons.apache.org/proper/commons-io/download_io.cgi

7.4 Dokumentation

Die Textdokumentation wurde mit dem Textverarbeitungsprogramm Microsoft® Word 2013 als Bestandteil von Microsoft Office Professional Plus 2013 erstellt.

Die Nassi-Schneidermann-Diagramme wurden mithilfe von EasyCODE 9.3 erstellt. <http://www.easycode.de/download/software/>

UML-Klassendiagramme wurden in Visual Paradigm Version 14.2 erstellt. <https://www.visual-paradigm.com/download/>

8 Abweichungen vom Konzept

Die Implementation entspricht im Großen und Ganzen des entworfenen Konzepts, es ist bei der Umsetzung allerdings zu kleineren Änderungen und vor allem zu Ergänzungen gekommen. Diese Änderungen werden hier dokumentiert.

8.1 Abweichungen und Ergänzungen des Algorithmus

Bei der Ausarbeitung des Konzepts war noch keine sinnvolle Vorbereitung auf die Iterationen modelliert. Es wird jetzt vor Beginn der Iterationsschleife zunächst eine Kräfteberechnung durchgeführt und auf Überschneidungen überprüft. Bei höheren Abstoßungskräften als Anziehungskräften werden die Kennwerte so lange halbiert bis global größere Anziehungskräfte wirken. Dies entspricht dem „auseinander schieben“ welches in der Aufgabenstellung empfohlen wurde. Im Zuge dieser Änderung musste der Kennwert der zunächst als Integer angedacht war in das Double Format überführt werden und wird jetzt als solcher im Staat-Attribut gespeichert. Außerdem wurden verschiedene Hilfsmethoden hinzugefügt die diese Berechnung erleichtern oder ermöglichen, so musste der Staat-Klasse eine Skalier Funktion hinzugefügt werden und in *Laenderberechnung* wird Verwendung von den bereits angedachten Methoden zur Berechnung aller Abstoßungs- und Anziehungskräfte eines Staats gemacht. Zur Summierung dieser Kräfte für alle Vektoren wurden in *Laenderberechnung* Hilfsmethoden hinzugefügt.

Diese Hilfsmethoden finden ebenfalls Verwendung für die Umsetzung der bis dahin noch fehlenden Qualitätsprüfung und in diesem Zusammenhang den Abbruchbedingungen der Iterationsschleife. Für die Abbruchbedingungen wurden ebenfalls zwei weitere Attribute hinzugefügt. Die maximale Anzahl von Iterationen nach denen die Schleife spätestens abbricht und der Toleranzwert der in der Verfahrensbeschreibung näher beschrieben wurde.

Zusätzlich wurde noch eine Methode zur Berechnung des jeweils sinnvollen Darstellungsbereichs einer Iteration hinzugefügt.

Der Algorithmus einer einzelnen Iteration musste insofern angepasst werden dass die Kräfte nicht jeweils häufig auf einen Staat wirken, sondern abhängig von der Anzahl der Nachbar, wie in der Verfahrensbeschreibung erläutert.

8.2 Abweichungen in der Klassenstruktur

Die Funktion zur Berechnung der Kräfte war zunächst als Methode von *Laenderberechnung* modelliert, allerdings hat sich herausgestellt dass es sinnvoller ist wenn diese Berechnung in *Staat* stattfindet da so kein externer Zugriff auf die Nachbarstaaten benötigt wird und unnötige Hilfsmethoden entfernt werden können.

Wie bereits in 8.1 angesprochen wurden verschiedene Klassen um neue Hilfsmethoden ergänzt.

Zusätzlich wurde die *Feedback*-Klasse entwickelt welche die Rückgabe über die Kommandozeilenschnittstelle verwaltet. Es wurde jeweils in *Laenderberechnung*, *Staat* und *Vektor* ein Attribut dieser Klasse erstellt und instanziert.

9 Zusammenfassung und Ausblick

Das entwickelte Programm dient zur Berechnung von Kartendarstellungen anhand gegebener Kennwerte, Lage- und Nachbarschaftsbeziehungen.

Es wird eine Eingabedatei mit den entsprechenden Werten eingelesen, diese Werte werden dann in eine sinnvolle Darstellung gebracht.

Bei der Darstellung wird vor allem darauf geachtet, dass ein ausgeglichenes Verhältnis zwischen möglichst wenig Überschneidungen und möglichst geringen Abständen zwischen Nachbarstaaten erreicht wird. Da dieses Verhältnis nur ein möglicher Lösungsansatz ist kann es sich in der Darstellung im Detail von ähnlichen Implementationen unterschieden.

Die Ergebnisse werden dann in einer Ausgabedatei ausgegeben. Das Format der Ausgabedatei entspricht dabei gültigen *gnuplot* Befehlen wodurch die erarbeiteten Ergebnisse grafisch Darstellbar sind.

Die Architektur basiert auf dem EVA-Prinzip der modularen Entwicklung. Durch diese Trennung der einzelnen Module und der zur Verfügung gestellten Schnittstellen für die Eingabe und Ausgabe ist das Programm leicht erweiterbar.

Anhand eines Testfalls wurde gezeigt dass der Algorithmus insbesondere in der Wahl des Abbruchkriteriums verbessertswürdig ist. Eine mögliche Lösung ist es zusätzlich zu dem globalen Kräfteverhältnis noch die lokalen Kräfte zwischen allen Nachbarn eines Staats zu vergleichen.

Darüber hinaus kann das hier entwickelte Programm mit einer eigenen graphischen Oberfläche für die direkte Darstellung der Karten erweitert werden.

10 Abbildungsverzeichnis

Abbildung 1: Beispiel einer Eingabedatei.....	4
Abbildung 2: Gerüst der Ausgabedatei.....	5
Abbildung 3: Struktogramm des Eingabe-Moduls.....	7
Abbildung 4: Struktogramm der Staat-Klasse	7
Abbildung 5: Struktogramm der Vektor-Klasse	8
Abbildung 6: Struktogramm der Laenderberechnung-Klasse	8
Abbildung 7: Struktogramm des Ausgabe-Moduls.....	9
Abbildung 8: Struktogramm der Feedback-Klasse.....	9
Abbildung 9: Klassenstruktur im Überblick	10
Abbildung 10: NS-Diagramm von berechneKartenDarstellung.....	11
Abbildung 11: NS-Diagramm von berechneKraft	11
Abbildung 12: NS-Diagramm von berechnelteration	12
Abbildung 13: NS-Diagramm von kraefteAnwenden.....	12
Abbildung 14: NS-Diagramm von berechneDarstellungsbereich	13
Abbildung 15: Ordnerstruktur	14