Abschlussprüfung Sommer 2022
Entwicklung eines Softwaresystems

Vorgelegt von:	Sven Bergmann
Vorgelegt am:	8. April 2022
Prüfungsnummer:	123456789
Ausbildungsbetrieb:	CAE GmbH

Inhaltsverzeichnis

1 Aufgabenbeschreibung

1.1 Analyse Problemstellung

Es soll ein Programm entwickelt werden, welches Länder mit staatenspezifischen Kennwerten in Form von Kreisen, proportional zur jeweiligen Größe des Kennwertes wiedergibt. Dabei muss darauf geachtet werden, dass die Abstoßungskräfte (zwei Kreise überschneiden sich) und die Anziehungskräfte (zwei Kreise berühren sich nicht, sind aber Nachbarn laut Vorgabe) präzise und in mehreren Iterationen so berechnet werden, dass sich am Ende keine Überschneidung mehr ergibt und die Nachbarschaften bestmöglich eingehalten werden.

1.2 Geeignete Programm- und Datenstruktur

Zuerst wird jeder Staat in einem Objekt der Klasse „Staat“ repräsentiert, welches einige Attribute bereitstellt, wie „identifier“, „kenngroesse“, „x“ und „y“. Der Längengrad wird dabei bei der Initialisierung direkt auf den x-Wert gemappt, während der Breitengrad auf den y-Wert gesetzt wird. Diese Staaten werden dann in eine Klasse „Landkarte“ gekapselt, die zudem noch die Kenngröße, Beziehungen unter den Staaten, Kräfte unter den Staaten, Anzahl der Iterationen und die Strategie enthält. Mit dem Objekt dieser Klasse wird dann auch gerechnet. Für die Eingabedaten existiert eine Klasse namens „FileLandkartenReader“, welche die vorgegebene Datei direkt in ein Landkartenobjekt lesen kann. Die Ausgabe erfolgt dann über die Klasse „GnuPlotWriter“, bei der wie der Name schon sagt, die Ausgabe in ein lesbares Format für das GnuPlot Programm[1] gebracht wird. Weiterhin existieren Helper Klassen, wie z. B. „Kreis“, „Punkt“ oder „Pair“, die nützliche, wiederverwendbare Methoden bereitstellen. Der Algorithmus wird über die Klasse „IStrategy“ implementiert um das Ganze austauschbar zu machen. Bisher existiert allerdings nur die Klasse „BruteForceStrategie“, welche eine Strategie implementiert.

1.3 Einlesen und Initialisieren der Daten

Da davon auszugehen ist, dass die Eingabedaten keine Fehler enthalten, wird die Eingabe vorher auch nicht geprüft. Die Datei fängt also mit einer Zeile an, worin die zu berücksichtigende Kenngröße steht. Diese Zeile wird als String eingelesen und als Attribut der Klasse „Landkarte“ gespeichert. Die zweite Zeile enthält einen Kommentar, welcher mit „#“ beginnt. Soweit es sich nicht besser aus den Beispielen erschließen lässt, ist nicht noch eine weitere Kommentarteile vorhanden. Danach folgt eine Liste aller zu berücksichtigenden Staaten, wobei in jeder Zeile ein Staat mit Kennzeichen, Kenngröße, Längen- und Breitengrad angegeben wird. Diese Staaten können dadurch direkt initialisiert und als Schlüssel in die Map „Beziehungen“ der Klasse „Landkarte“ eingefügt werden. Sind alle Staaten eingelesen, folgt eine Kommentarzeile „Nachbarschaften“. Danach folgen alle Nachbarschaftsbeziehungen in der Form: „<Staat>: <Liste aller Nachbarn getrennt mit \s>“. Diese werden in das Attribut „Beziehungen“ eingelesen, wobei pro Zeile zuerst nach

[1] `http://www.gnuplot.info/`

dem Schlüssel gesucht wird, und danach jeweils der Staat dem Set hinzugefügt wird. Eine Darstellung als Nassi-Shneiderman Diagram ist in Abbildung 7 auf Seite 16 vorhanden.

1.4 Entwurf des Algorithmus

Der Algorithmus orientiert sich sehr an der Angabe des Verlaufsdiagramms in der Aufgabenstellung. Bevor irgendetwas gerechnet wird, werden alle Kenngrößen nach dem Einlesen normalisiert, indem jeder Kennwert durch den größten Kennwert geteilt wird. So wird dem Problem aus dem Weg gegangen, dass schon direkt am Anfang Kreise überlappen. Der Algorithmus startet nun und berechnet bei jedem Durchlauf zuerst die Kräfte der sich überlappenden Kreise[2]. Sind also die Staaten keine direkten Nachbarn voneinander, überlappen sich aber die von den Kenngrößen und Mittelpunktskoordinaten aufgespannten Kreise, wird dem Attribut „kraefte“ des Objektes „Landkarte“ eine Kraft hinzugefügt. Danach werden alle direkten Nachbarn jedes Staates betrachtet. Für jeden Nachbarn wird mit einer Abstandsberechnung der Kreis-Klasse berechnet, ob die Kraft negativ (abstoßend), oder positiv (anziehend) sein muss. Ist der Algorithmus damit fertig, wird nun eine HashMap angelegt, welche zu einem Schlüssel (Staaten Identifizierer) jeweils ein HashSet speichern soll, zu dem alle Punkte hinzugefügt werden, wohin sich der Kreis verschieben soll. Mit dem letzten Schritt iteriert der Algorithmus durch diese HashMap, berechnet den Mittelwert der Punkte für jeden Staat und verschiebt die Staaten dementsprechend.

1.5 Ausgabe gemäß Aufgabenstellung

Die Ausgabe soll in einer Form dargestellt werden, welche von GnuPlot lesbar ist. Hierfür muss zuerst der x-Bereich, sowie der y-Bereich berechnet werden. Dafür wird durch die Staaten durchiteriert und zusammen mit dem Radius der Kreise der kleinste, bzw. größte x, bzw. y Wert berechnet. Ist nun $\mid x_{max} - x_{min} \mid \neq \mid y_{max} - y_{min} \mid$, so wird der kleinere Bereich um jeweils die Hälfte des Überschusses links und rechts erweitert, sodass GnuPlot die Kreise ordentlich ausgibt und diese nicht verzerrt. Der Name der Kenngröße, sowie die Anzahl der Iterationen wird dann aus dem Attribut des Objektes „Landkarte“ ausgelesen. Die Liste kann danach zusammen mit allen anderen Attributen geschrieben werden. Der Workflow hierfür ist ebenfalls in einem Nassi-Shneiderman Diagram in Abbildung 8 auf Seite 17 dargestellt.

2 Objektorientierter Entwurf

2.1 Klassendiagramme

Die UML-Klassendiagramme befinden sich im Anhang. Auf der obersten Ebene befindet sich die „Main“ Klasse zusammen mit zwei anderen Paketen[3]. Im darunterliegenden Paket

[2] s. Abbildung 9 auf Seite 18
[3] s. Abbildung 1 auf Seite 10

„models“[4] befinden sich die Klassen „Landkarte“ und „Staat“. Das Paket „utils“[5] beinhaltet neben der Klasse „Pair“ und „LogOption“ auch noch das Paket „algorithm“[6], das Paket „io“[7] und das Paket „la“[8].

2.2 Pakete

2.2.1 Utils

Das Paket „com.cae.de.utils“ beinhaltet weitere Pakete, welche Klassen beinhalten, die in irgendeiner Art zum Oberbegriff „Utilities“ oder „Werkzeuge“ gehören. Unter anderem existiert hier die Record Klasse „Pair“ welche zum erstellen eines „immutable Pairs“ nützlich sein kann, da keine Pair-Klasse mehr in Java 17 existiert. Des Weiteren wird die Klasse „LogOption“ in der Main Methode genutzt um die Option für das Logging festlegen zu können. Der Nutzer hat dabei die Möglichkeit die Logging Ausgabe zu modifizieren.

IO Wie der Name schon sagt, ist dies das Paket für die Ein- und Ausgabe von Daten. Es existieren vor allem die Interfaces „IReader“ und „IWriter“, welche beide generisch geschrieben sind, sodass in den implementierenden Klassen angegeben werden kann, welche Objekte eingelesen oder geschrieben werden. Zudem existieren hier noch implementierende Writer und Reader, wie „GnuPlotWriter“ und „FileLandkartenLeser“.

Algorithms Das Paket Algorithms ist auch selbst sprechend und beinhaltet Klassen, welche Strategien implementieren. Vor allem wird hier darauf geachtet, dass durch die Implementierung des Interfaces „IStrategy“ die Algorithmen austauschbar gemacht werden.

LA In „la“ oder „lineare Algebra“ befinden sich die Record Klassen „Punkt“ und „Kreis“, welche beide Methoden der linearen Algebra bereitstellen um beispielsweise Abstandsberechnungen durchführen zu können.

2.2.2 Models

Im Paket „models“ befinden sich nur die zwei Klassen „Landkarte“ und „Staat“. Da diese Klassen fast nur eine Datenhaltung bereitstellen, befinden sie sich hier.

[4] s. Abbildung 2 auf Seite 11
[5] s. Abbildung 3 auf Seite 12
[6] s. Abbildung 4 auf Seite 12
[7] s. Abbildung 5 auf Seite 13
[8] s. Abbildung 6 auf Seite 14

3 Änderungen zur schriftlichen Ausarbeitung

3.1 Umbenennungen

Es sind fast alle der Bennenungen so wie in der schriftlichen Ausarbeitung geblieben, es wurden nur alle Bennenungen an das Deutsche angepasst. Die Java-spezifischen Benennungen wie „z. B. getAttribut oder setAttribut“ sind geblieben. Zudem wurden natürlich noch die Umlaute umschrieben.

3.2 Modifikation der Klassenstruktur

In der schriftlichen Ausarbeitung war ni die Rede von einem Interface namens „IStrategy“. Dies ist hinzugenommen, um den zu implementierenden Algorithmus austauschbar zu machen. Die Utils Klassen „Pair“ und „Punkt“ existierten ebenfalls noch nicht und stellten sich zusammen mit der Implementierung der Klasse „Kreis“ und „Main“ als nützliche Klassen dar. Ansonsten sollte sich nichts geändert haben.

3.3 Modifikation der Logik

Während des Einlesens wurden zwei oder mehr Listen angelegt, obwohl die Klasse „Landkarte“ nur genau eine HashMap mit allen Beziehungen braucht, um die Staaten zu speichern. Direkt nach dem Einlesen wird die Kenngröß jedes Staates normalisiert, das heißt $\nabla Staaten : \frac{staat.kenngroesse}{staaten.maximaleKenngroesse}$.Weiterhin wurde der Algorithmus in der Art angepasst, dass nun eine Art Qualitätskontrolle stattfindet, welche den derzeitigen Abstand aller Nachbarstaaten berechnet und falls dieser geringer ist, als der vorher gespeicherte Abstand, diesen ersetzt und zudem die HashMap aller Beziehungen speichert. Diese HashMap wird dann am Ende genommen, um die Eigentliche zu überschreiben und dadurch wird durch das Kommandozeilenargument, welches die Iterationen angibt nur eine maximale Anzahl an Iterationen festgelegt. Es kann somit beispielsweise passieren, dass bei einer Angabe von „-i 500“ die Anzahl 460 an Iterationen herauskommt, da dort der minimalste Abstand der Nachbarstaaten ist. Es fand zudem eine Veränderung der Abstandsberechnung statt, da in irgendeiner Art dargestellt werden musste, dass ein Kreis innerhalb eines anderen liegt. Des Weiteren wird der Staat mit den meisten Nachbarn von Verschiebungen ausgenommen, da es so einen Fixpunkt gibt, um den sich die anderen Staaten herum bewegen können.

4 Benutzeranleitung

Generell befindet sich die Gesamtdokumentation der Klassen und Methoden als javadoc im „docs“ Ordner.

4.1 Ordnerstruktur

An sich sollten die fertig gebaute „.jar“ Datei zusammen mit dem ausführbaren Python Skript in einem Ordner liegen. Die Testbeispiele sollten ebenfalls auf der gleichen Ebene

in einem Ordner vorhanden sein, wobei der Ordnername in beim Programmstart oder in dem Skript angegeben werden kann. Der Ordner für alle Ausgabe Dateien wird dann ebenso in dieser Ebene erstellt. Die Ausgabedateien werden pro Eingabedatei jeweils in eine Datei namens „<Name der Eingabedatei>_out.txt“ im output Ordner gespeichert, welcher entweder über die Programmzeile vorher angegeben wird, oder einfach „output“ heißt.

4.2 Benötigte Programme

Für die Ausführung der „jar“ Datei benötigt der Zielrechner zwingend eine Installation des Open JDK 17[9], da das Java-Sprachlevel auf 17 gesetzt wurde. Um das Python Skript auszuführen wird eine Python Installation gebraucht, wobei ich die Version 3.10[10] benutzt habe. Um das Projekt zu bauen, wird „gradle“[11] genutzt.

4.3 Ausführen als Kommandozeilenprogramm

Es gibt zwei Wege das Programm auszuführen, einmal direkt über Java und einmal über das Python Skript, welches intern die „jar“ Datei aufruft.

Das Programm nimmt Argumente entgegen, welche sind:

- -inputfolder {„Name des Ordners mit den Eingabedateien“}
- -outputfolder {„Name des Ordners, wohin die Ausgabedateien geschrieben werden“}
- -log {„true“ oder „false“ oder „file“}
- -loglvl {„warning“ oder „info“}

Die Namen der Ordner werden immer relativ zum derzeitigen Ordnerpfad ausgewertet. Ohne den Parameter „-inputfolder“ wird das Programm den Ordner „input“ auf der gleichen Ebene suchen und wenn dieser vorhanden ist, das Programm ausführen, ansonsten das Programm direkt beenden. Der Parameter „-outputfolder“ muss nicht zwingend angegeben werden, da ohne diese Angabe alle Ausgabedaten in den Ordner „output“ geschrieben werden. Die Option „-log“ gibt an, ob der Benutzer Loggingausgaben in der Kommandozeile haben möchte oder nicht, der default Wert hierfür ist „true“, also Loggingausgaben in der Konsole. Falls die Option „file“ angeben wurde, werden alle Logs in eine Datei „log.txt“ auf der gleichen Ebene geschrieben. Weiterhin wird mit der Option „-loglvl“ angegeben, ab welchem Level geloggt werden soll. Die Voreinstellung ist hier „Level.all“.

[9] `https://www.oracle.com/java/technologies/downloads/#java17`
[10] `https://www.python.org/downloads/release/python-3100/`
[11] `https://gradle.org/`

4.3.1 Ausführen der JAR

Um die „.jar“ ausführen zu können, muss zumindest unter Windows der Pfad zum JDK 17 in den Umgebungsvariablen gesetzt sein. Mit einem Doppelklick auf die Datei wird der Code mit Standardargumenten ausgeführt. Ansonsten könnte der Benutzer auch über die Kommandozeile gehen und die Programmargumente selbst setzen. Das würde dann beispielsweise so aussehen:

```
java -jar IHK_Abschlusspruefung.jar -inputfolder input -outputfolder
    output -log false
```

4.3.2 Ausführen des Python-Skripts

Der Benutzer kann zusätzlich auch noch das Python-Skript „execute_gro_pro.py“ ausführen, um das Programm mit den Standardargumenten zu starten. Hierfür ist eine Installation von Python notwendig, sowie die Verlinkung zur Umgebungsvariablen. Der Kommandozeilencode sieht dann beispielsweise so aus:

```
python execute_gro_pro.py
```

5 Beschreibung, Begründung und Diskussion ausgewählter Beispiele

5.1 Black-Box-Test

Da die folgenden Beispiele, welche auch in der Aufgabenstellung genutzt werden, nicht darauf abziehlen, den Programmcode im Einzelnen zu verstehen und zu testen, fallen diese unter den Bereich der Black-Box-Tests. Des Weiteren werden keine Zwischenschritte geprüft, sondern einzig die Eingabe, sowie die ungefähre Ausgabe vorgegeben.

5.1.1 Beispiel 1: Fläche von 10 Staaten

Das erste Beispiel der Aufgabenstellung beschreibt die Fläche von 10 Staaten in Europa und deren Beziehungen. Hierbei habe ich, wie am Anfang erklärt, auch die Kenngröße normalisiert, da die Kreise der Staaten ansonsten direkt überlappen würden. Das Beispiel war dahingehend gut gewählt, da man die ungefähre Position der Staaten im Hinterkopf hat und dadurch besser evaluieren kann, wie gut oder schlecht, der selbst geschriebene Algorithmus ist. Bisher konnte ich weder mit 100 Iterationen, noch mit einer anderen Zahl an Iterationen das Beispiel reproduzieren.

Das Beispiel ist in der Datei „beispiel1.txt“ im Ordner „input“ zu finden.

5.1.2 Beispiel 2: Bierkonsum von 10 Staaten

Der Bierkonsum der Staaten wird in Beispiel 2 beschrieben, was daher interessant ist, dass alle Nachbarschaftsbeziehungen wie in Beispiel 1 dargestellt werden müssen, aber die Größe der Kreise vollkommen unterschiedlich sind. Es kann also nun passieren, dass

der Algorithmus mit Beispiel 1 besser oder schlechter funktioniert als mit Beispiel 2. Allerdings konnte hier wiederum weder mit 100 noch mit einer anderen Zahl an Iterationen das Beispiel reproduziert werden.

Das Beispiel ist in der Datei „beispiel2.txt“ im Ordner „input“ zu finden.

5.1.3 Beispiel 3: Fläche von 24 Staaten

Die Fläche von 24 Staaten ist wiederum ähnlich zum Beispiel 1 nur mit mehr Nachbarn. Das heißt alle Kenngrößen bleiben gleich und theoretisch sollte auch die Anordnung der 10 gleichen Staaten gleich bleiben, allerdings kann es durch die Abstoßungs- und Anziehungskräfte passieren, dass sich die Konstellation auch ändert.

Das Beispiel ist in der Datei „beispiel3.txt“ im Ordner „input“ zu finden.

5.1.4 Beispiel 4: 2 Staaten zum Testen

In diesem Beispiel habe ich zwei Staaten „1“ und „2“ zum Testen angelegt. Normalerweise sollten sich die Kreise direkt nach der ersten Iteration richtig ausgerichtet haben.

Das Beispiel ist in der Datei „beispiel4.txt“ im Ordner „input“ zu finden.

6 Zusammenfassung und Ausblick

In dieser Aufgabe mussten Kenngrößen für Länder mit geografischen Kennwerten dargestellt werden. Das Ziel dabei war, die geografische Anordnung und Nachbarschaftsbeziehungen bei verschieden großen Werten möglichst genau beizubehalten. Hierfür mussten Funktionen definiert werden, welche die Anziehungs- und Abstoßungskräfte der Kreise berechnen und diese dementsprechend verschieben. Mit einer festen Anzahl an Iterationen funktioniert der Algorithmus leider nicht, da dieser durch die Kräfte divergiert, das heißt die Kreise stoßen sich immer wieder ab und ziehen sich zusammen. Durch ein wie auch immer geartetes Qualitätskriterium kann man dieses Problem mehr oder weniger lösen, allerdings ist es nicht ganz eindeutig, wie dieses Kriterium aussehen soll. In meiner Lösung habe ich den Gesamtabstand der Nachbarstaaten voneinander genommen und beim minimalen Abstand die Beziehungen ausgegeben.

In einer Verbesserung des Programms könnte noch eingebaut werden, dass eine grafische Anzeige vorhanden ist, welche nicht die letzte Iteration anzeigt, sondern durch alle Iterationen durchläuft und jede einzeln anzeigt. Hiermit könnte dann eventuell auch der Algorithmus, oder die Abbruchbedingung verbessert werden, da grafisch genau dargestellt wird, was eigentlich passiert.

A UML-Diagramme

A.1 Klassendiagramme

Abbildung 1: Main UML Diagramm

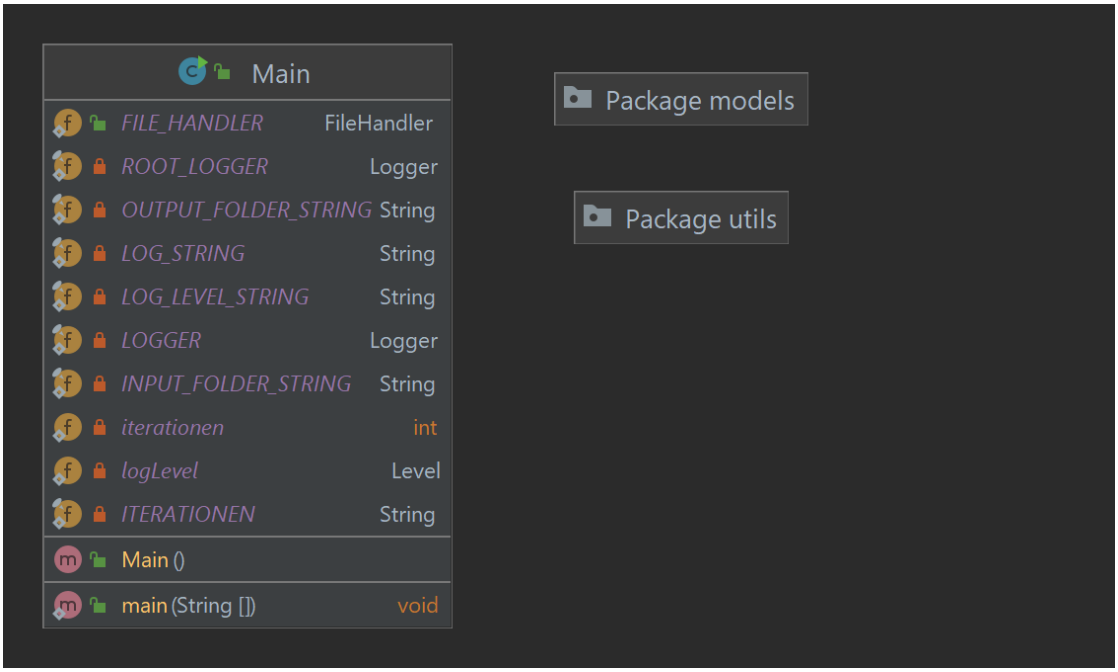


Abbildung 2: Paket models

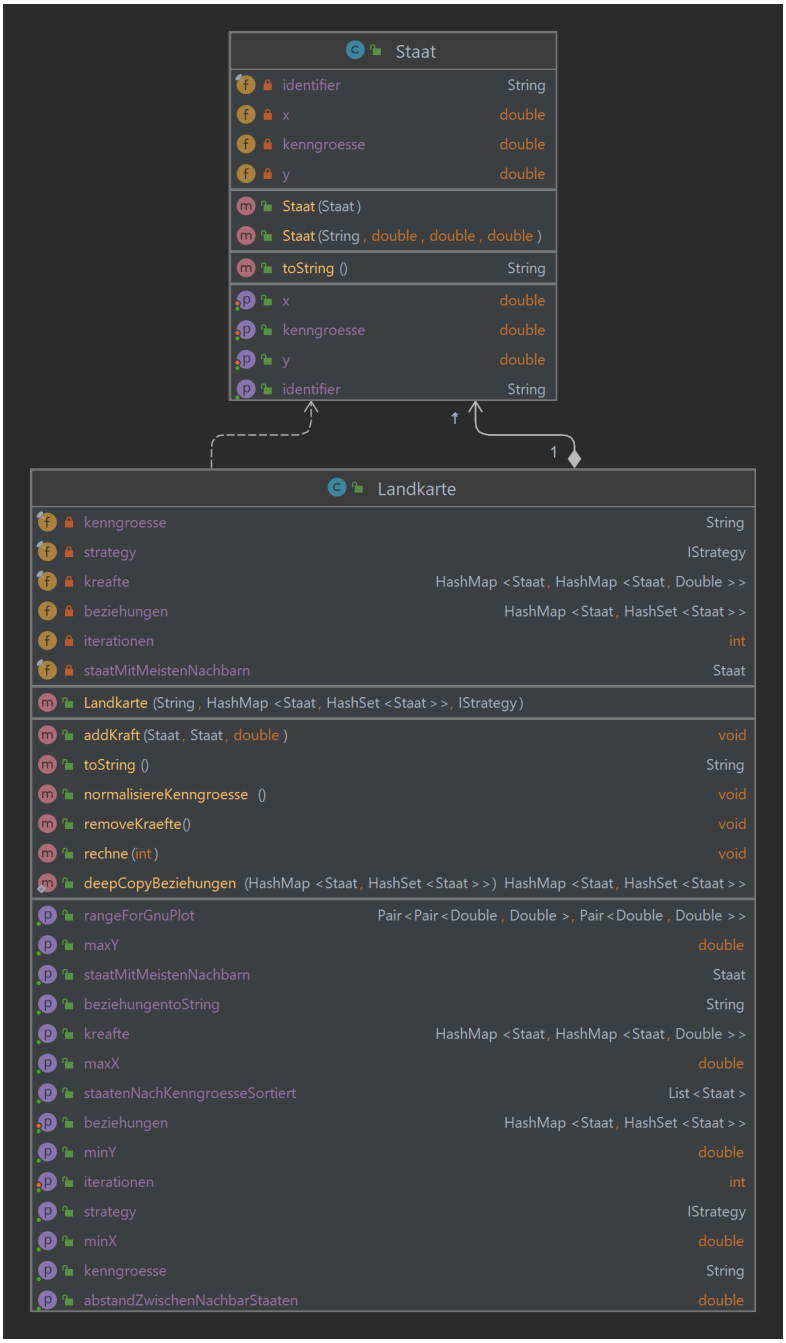


Abbildung 3: Paket utils

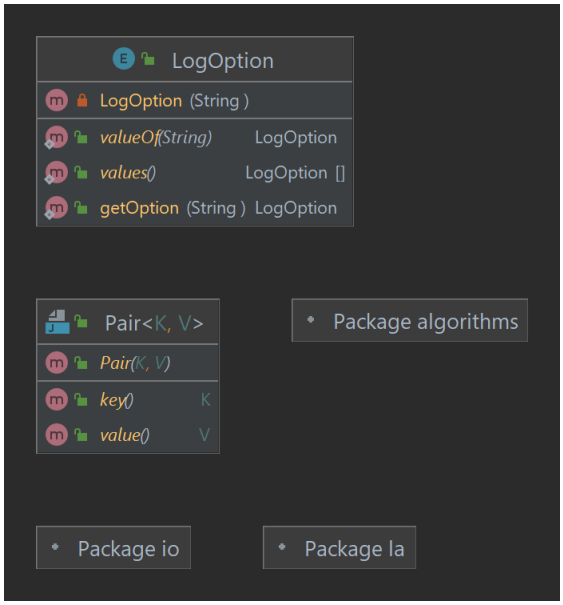


Abbildung 4: Paket algorithms

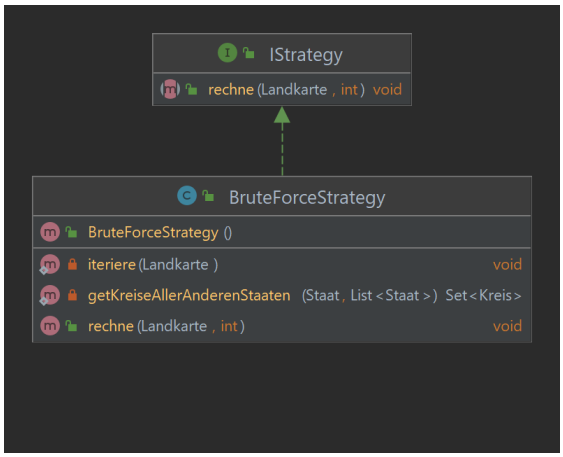


Abbildung 5: Paket io

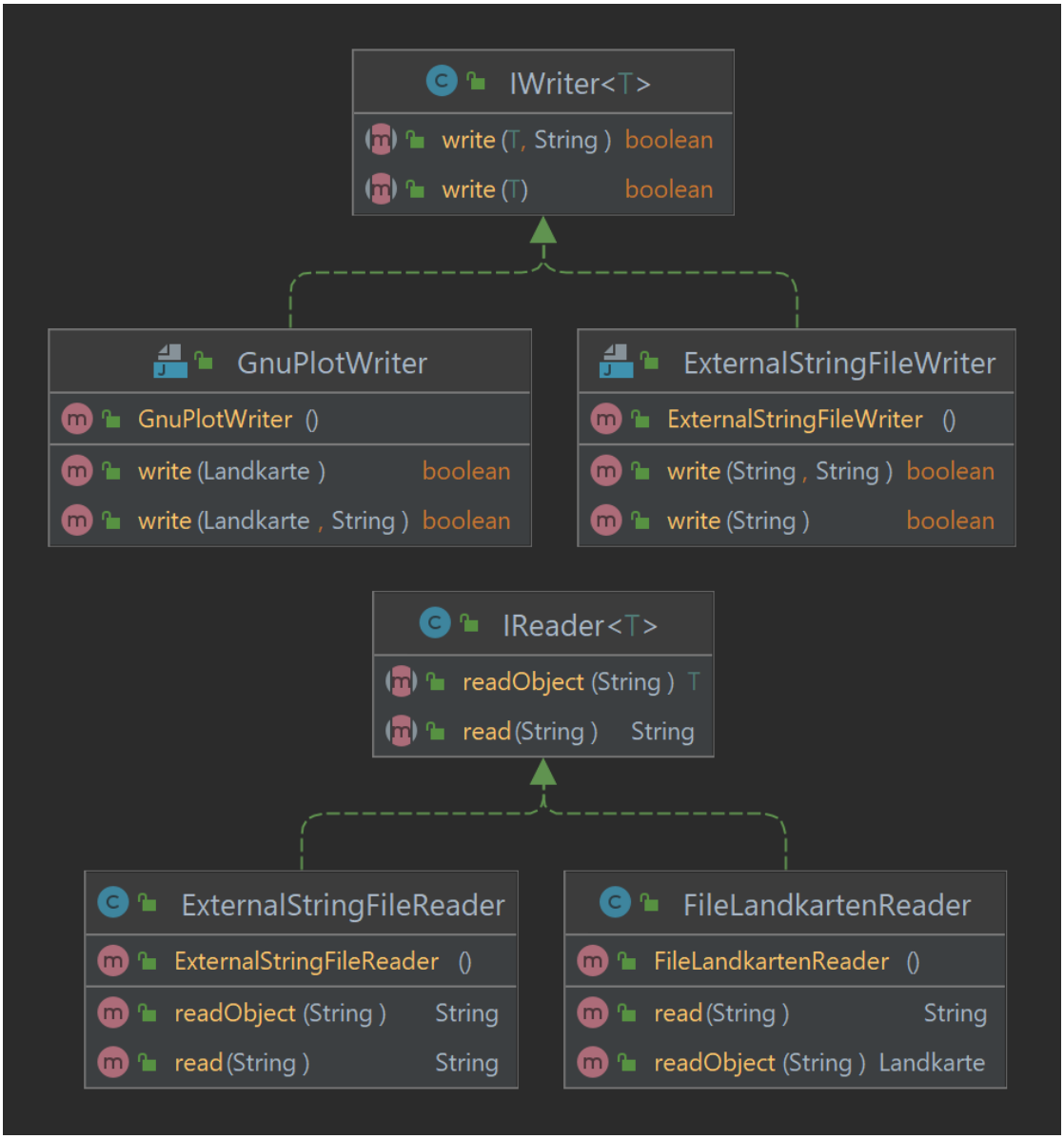
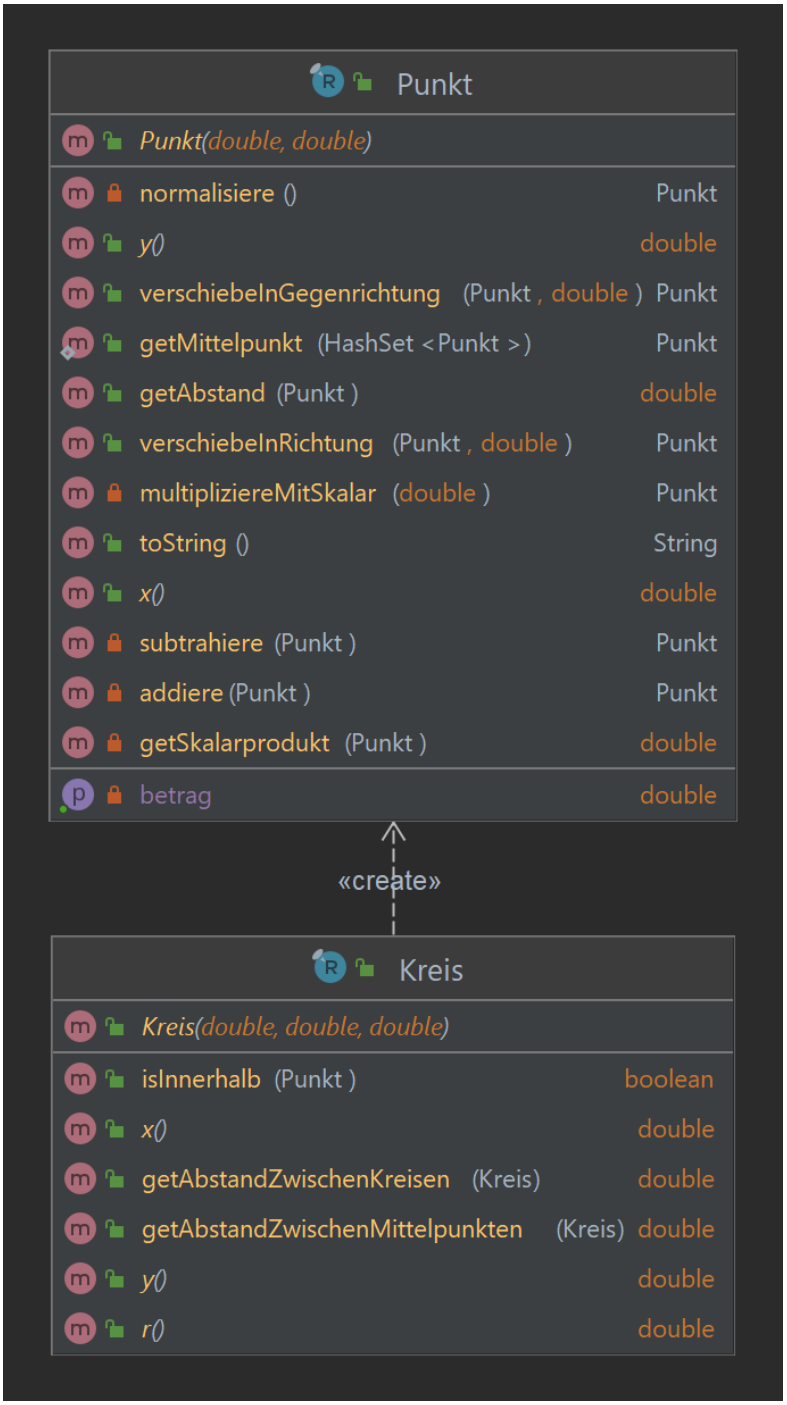


Abbildung 6: Paket la



A.2 Nassi-Shneidermann Diagramme

Abbildung 7: Einlesen der Landkarte

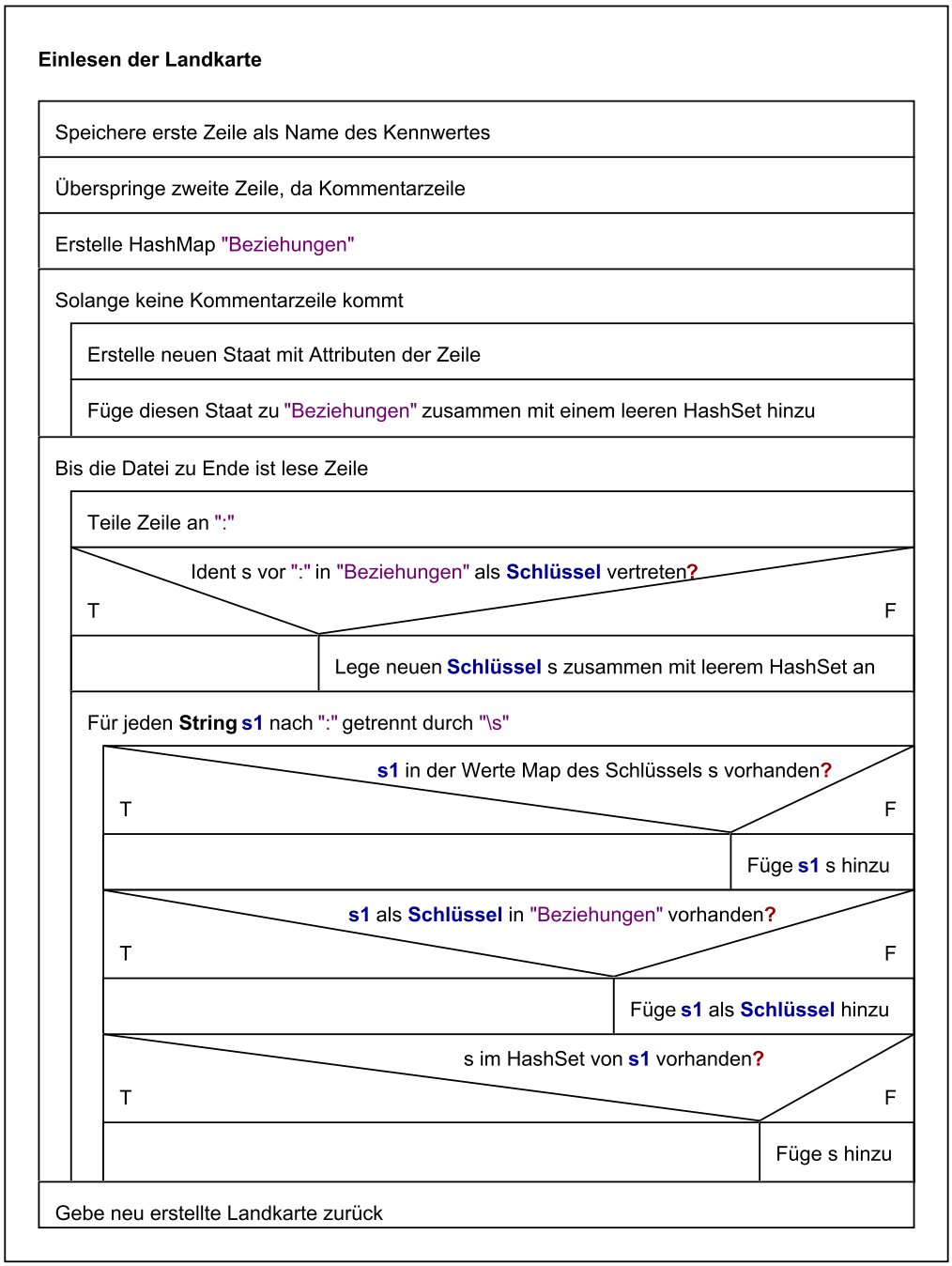


Abbildung 8: Ausgabe der Landkarte

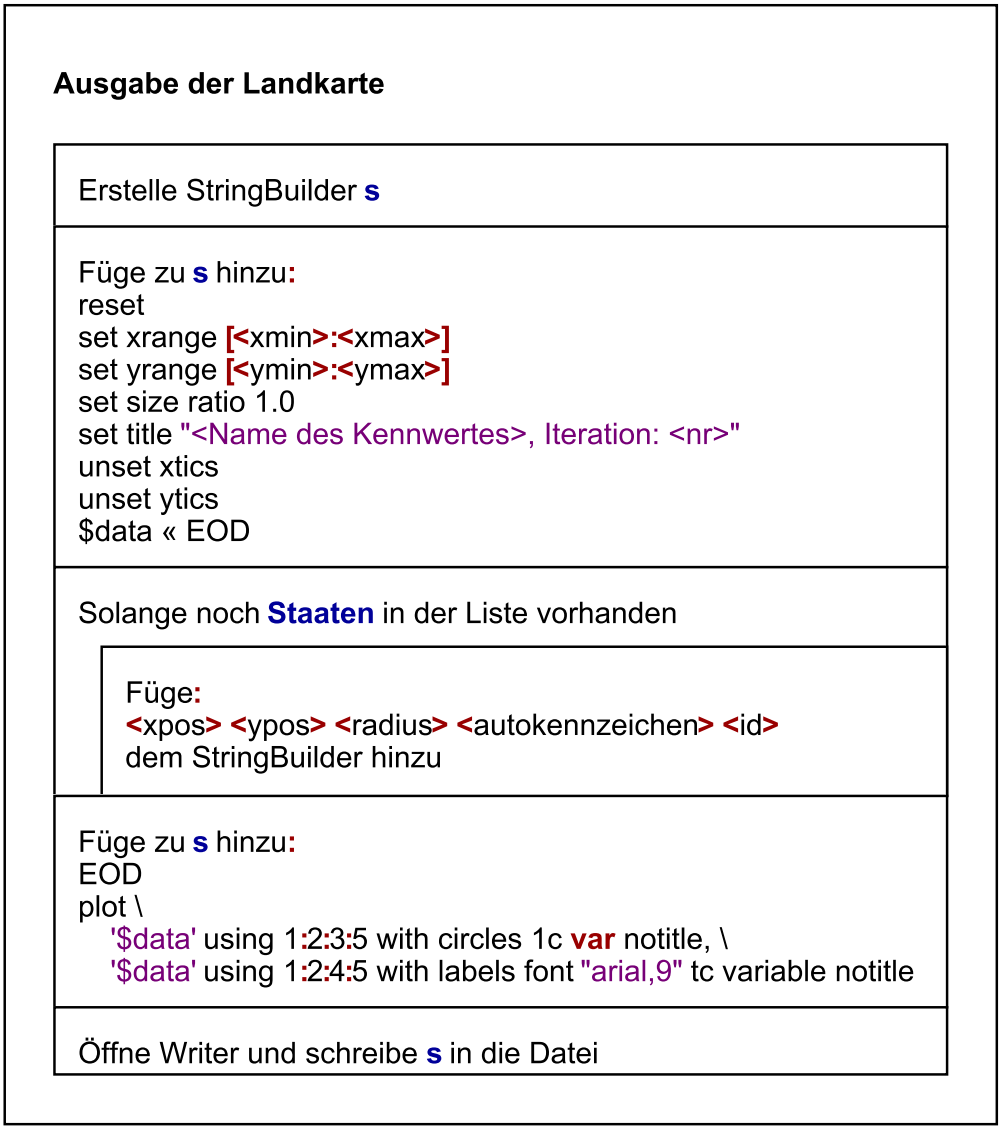
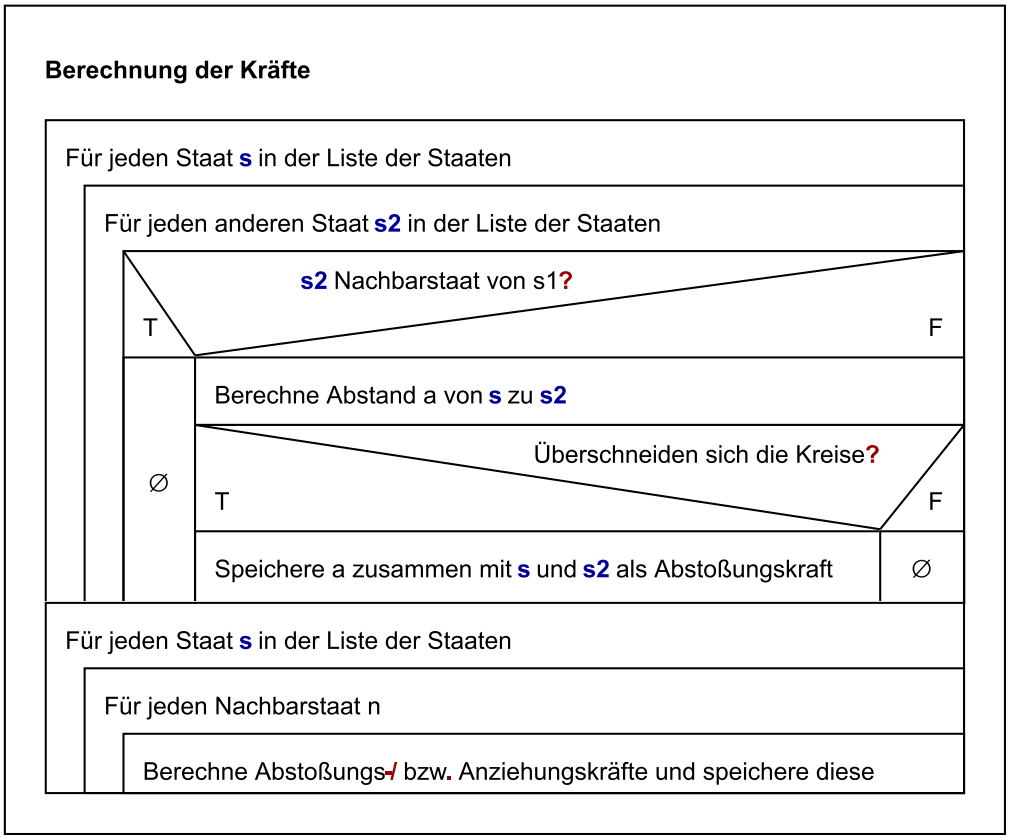


Abbildung 9: Berechnung der Kräfte



B Selbstständigkeitserklärung

Selbstständigkeitserklärung

ERKLÄRUNG

Hiermit erkläre ich, dass ich die vorgelegte

◯ Hausarbeit
◯ Bachelorarbeit
◯ Masterarbeit
◯ Staatsarbeit

selbstständig verfasst und - einschließlich eventuell beigefügter Abbildungen und Skizzen - keine anderen als die im Literaturverzeichnis angegebenen Quellen, Darstellungen und Hilfsmittel benutzt habe. Dies gilt in gleicher Weise für gedruckte Quellen wie für Quellen aus dem Internet.

Ich habe alle Passagen und Sätze der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, in jedem einzelnen Fall unter genauer Angabe der Stelle ihrer Herkunft (Quelle, Seitenangabe bzw. entsprechende Spezifizierung) deutlich als Entlehnung gekennzeichnet. Außerdem erkläre ich, dass die vorgelegte Arbeit zuvor weder von mir noch - soweit mir bekannt ist - von einer anderen Person an dieser oder einer anderen Universität eingereicht wurde. Eine Publikation der vorliegenden Arbeit ist nur mit Zustimmung des Dozenten möglich.

Mir ist bekannt, dass Zuwiderhandlungen gegen diese Erklärung eine Benotung der Arbeit mit der Note "nicht ausreichend" zur Folge haben. Ich weiß, dass Verletzungen des Urheberrechts sowie Betrugsversuche strafrechtlich verfolgt werden können und dass, wer vorsätzlich gegen eine die Täuschung betreffende Regelung verstößt, ordnungswidrig handelt. Die Ordnungswidrigkeit kann mit einer Geldbuße bis zu 50.000 Euro geahndet werden. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann außerdem eine Exmatrikulation erfolgen.

_____ _____

(Ort, Datum) (Unterschrift)