Entwurf

Praxis der Softwareentwicklung

Entwicklung einer Software zur Berechnung der Mandatsverteilung im Deutschen Bundestag

Gruppe 1

Philipp Löwer, Anton Mehlmann, Manuel Olk, Enes Ördek,
Simon Schürg, Nick Vlasoff

WS 2013 / 14

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Einleitung . . . 1
1.2 Notationshinweise . . . 1

2 Systemmodell **2**
2.1 Paketdiagramm . . . 2

3 Klassendiagramm **3**
3.1 GUI . . . 3
3.2 Datenhaltung . . . 6
3.3 Steuerung . . . 9
3.4 Import/ Export . . . 11
3.5 Wahlgenerierung . . . 13
3.6 Negatives Stimmgewicht simulieren . . . 15
3.7 Chronik . . . 17
3.8 Mandatsrechner . . . 18
3.9 Wahlvergleich . . . 20
3.10 Meldung . . . 21

4 Sequenzdiagramme **22**
4.1 Berechnung der Sitzverteilung . . . 22
4.2 DeepCopy . . . 23
4.3 Import . . . 25
4.4 Wahlgenerierung . . . 26
4.5 Negatives Stimmgewicht . . . 28
4.6 Chronik . . . 29
4.6.1 Veränderung an den Stimmen . . . 29
4.6.2 Restaurieren einer Stimme . . . 30
4.7 GUI . . . 31
4.7.1 Aktualisierung . . . 31
4.7.2 Stimmenänderung . . . 33

5 Implementierungsphasen-Zeitplan **34**

1 Einleitung

1.1 Einleitung

Dieses Dokument beschreibt den Entwurf der im Pflichtenheft spezifizierten Software zur Berechnung der Mandatsverteilung im Deutschen Bundestag.
Anhand verschiedener Diagramme, im Speziellen einem Klassendiagramm, werden die Architektur, die Komponenten, die Module und die einzelnen Klassen inklusive ihrer Schnittstellen und ihrer Attribute erläutert.
Des weiteren werden Entwurfsentscheidungen, Entwurfsdetails und die verwendeten Entwurfsmuster erläutert sowie zentrale Abläufe im Programm mit Hilfe von Sequenzdiagrammen visualisiert.
Abschließend wird die Zeitplanung der Implementierung und die zugehörigen Hauptverantwortlichen in einem Gantt-Diagramm dargestellt.

1.2 Notationshinweise

`Klassennamen` werden in der Erklärung des Klassendiagramms textuell hervorgehoben indem sie **fett** und in einer anderen Schriftart geschrieben werden.

`Methodennamen` werden in der Erklärung des Klassendiagramms textuell hervorgehoben indem sie *kursiv* und in einer anderen Schriftart geschrieben werden.

Außerdem wird Bundestagswahl im gesamten Entwurfsdokument durch BTW abgekürzt.

2 Systemmodell

2.1 Paketdiagramm

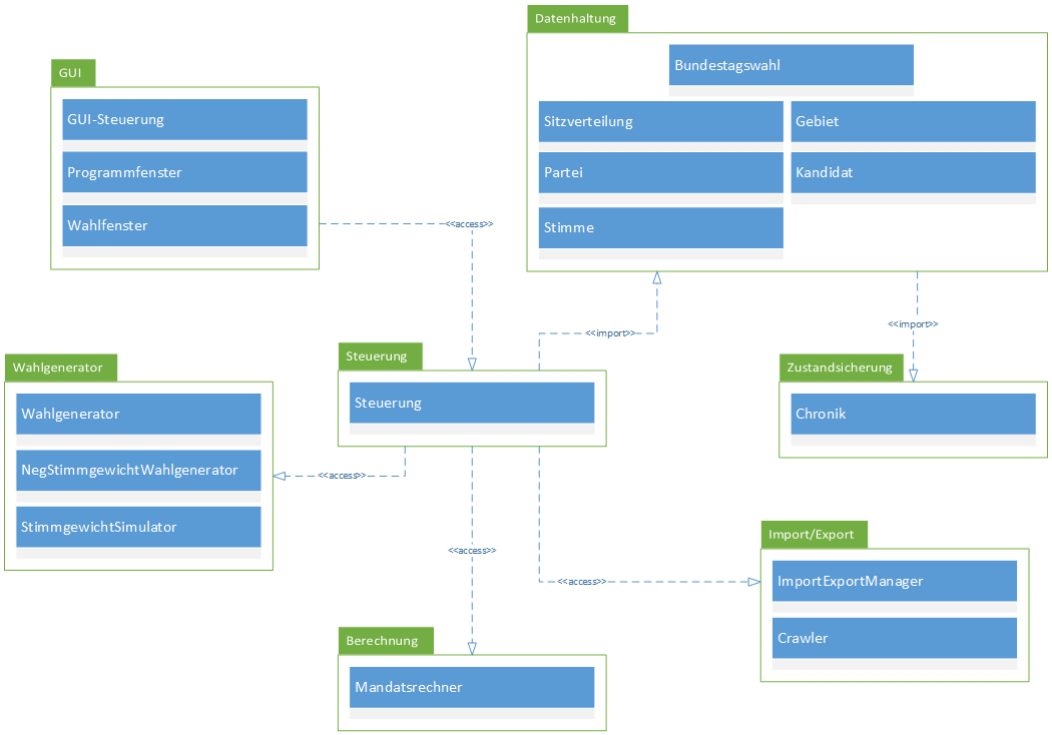

Abbildung 1: Paketdiagramm des Klassendiagramms

Durch das Paketdiagramms wird eine klar definierte Ansicht auf die Struktur und den Aufbau des modellierten System gewährleistet. Dabei wird insbesondere auf Paketübergreifende Zugriffe eingegangen. Zu erkennen ist, dass die Steuerung in dem Entwurf des vorliegenden Programms eine zentrale Position einnimmt.

3 Klassendiagramm

3.1 GUI

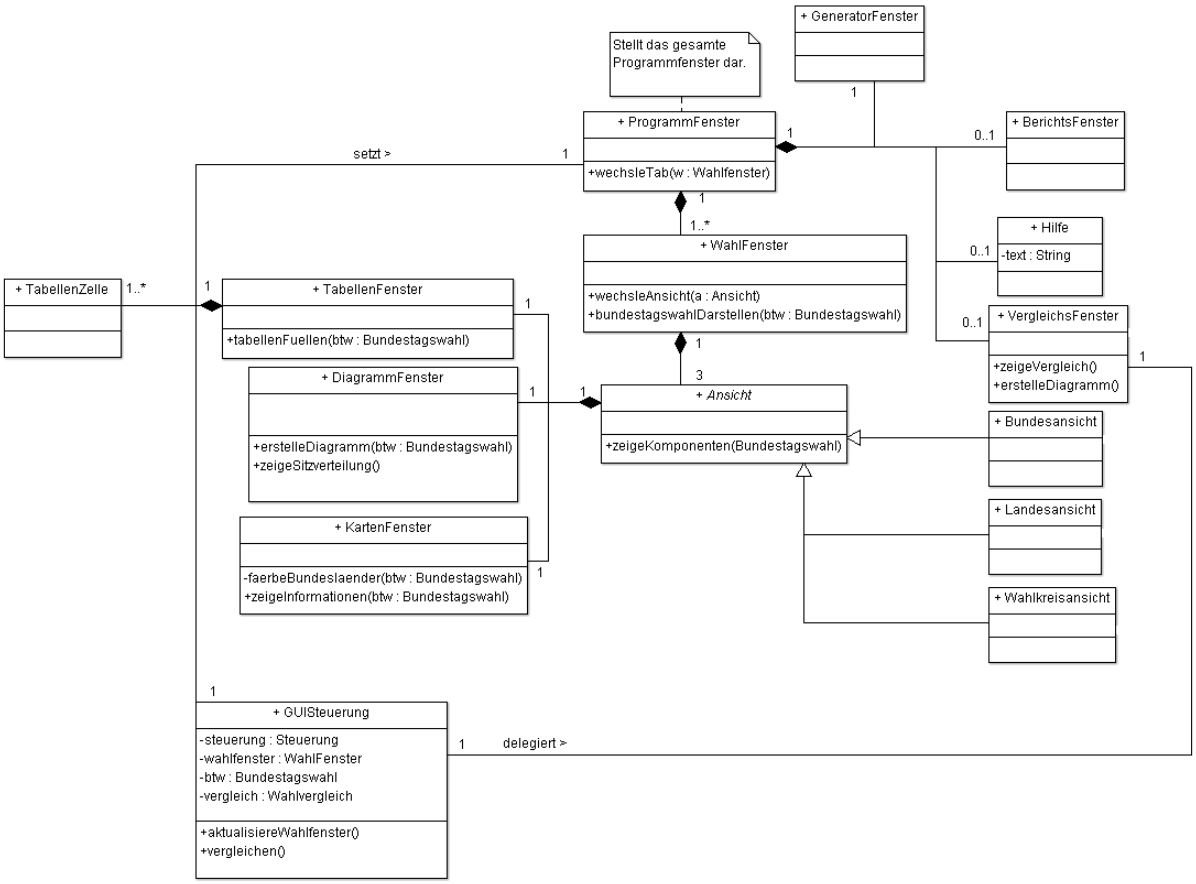

Abbildung 2: GUI Komponente

Zum GUI-Teil gehören alle Klassen, die dazu beitragen, das Datenmodell bestmöglich auf der grafischen Benutzeroberfläche darzustellen. Im folgenden werden wichtige Klasse der GUI erläutert.

`ProgrammFenster`

Das **`ProgrammFenster`** repräsentiert das gesamte Programm auf der grafischen Benutzeroberfläche.

`wechsleTab(w : WahlFenster)`

Diese Funktion wird beim Wechsel von Tabs aufgerufen. Dabei wird in der Steuerung das Attribut "aktuelleBTW" verändert auf die zu w zugeordnete *`BTW`*. Dies führt dazu, dass einige Funktionen in **`Steuerung`** wie *`setzeZurueck()`* ohne zusätzliche Parameter mit der Bundestagswahl des aktuell offenen Tabs arbeiten.

VergleichsFenster

Das **VergleichsFenster** stellt zwei **BTW**-Objekte in einem Fenster nebeneinander dar, sodass beide direkt vergleicht werden können.

zeigeVergleich()
Das **WahlBVergleich**s-Objekt der **GUISteuerung** wird in dem Fenster dargestellt.

erstelleDiagramm()
Erzeugt Diagramme unter den beiden Bundestagswahlen, welche die Sitzverteilungen anzeigen.

WahlFenster

Das **WahlFenster** repräsentiert eine einzelne **BTW**. In einem **ProgrammFenster** sind mehrere **WahlFenster** möglich, die in Form von Tabs dargestellt werden.

wechsleAnsicht(a : Ansicht)
Wechselt die Ansicht in eine andere gewünschte Ansicht.

bundestagswahlDarstellen(btw : BTW)
Verteilt den drei Ansichten die Arbeit, das **BTW**-Objekt darzustellen.

Ansicht

Die **Ansicht** ist eine abstrakte Klasse. Diese kann drei Arten von Ansichten sein, **Bundes-**, **Landes-** und **Wahlkreisansicht**. Diese werden im Folgenden nicht näher erläutert.

zeigeKomponenten(btw : BTW)
In jeder der drei Ansichten wird die Darstellung, des **BTW**-Objekts auf die drei Fenster aufgeteilt.

TabellenFenster

Das **TabellenFenster** korrespondiert zu dem im Pflichtenheft beschriebenen Tabellenfenster. Es stellt Teile der **BTW**-Klasse dar.

tabellenFuellen(btw : BTW)
Füllt das TabellenFenster mit den notwendigen Daten des **BTW**-Objekts.

DiagrammFenster

Das **DiagrammFenster** korrespondiert zu dem im Pflichtenheft beschriebenen Diagrammfenster. Je nach **Ansicht** wird dort die **Sitzverteilung**, prozentuale Anzahl der **Zweit-** oder **Erststimmen** angezeigt.

erstelleDiagramm(btw : Bundestagswahl)
Erstellt ein Diagramm auf Grund der Daten der **BTW** btw.

zeigeSitzverteilung()
Öffnet ein BerichtsFenster, in dem die Sitze der Verteilung näher erläutert werden.

KartenFenster

Das **KartenFenster** korrespondiert zu dem im Pflichtenheft beschriebenen Kartenfenster. Hier wird eine Liste der **Bundesländ**er und **Wahlkreise** und, wenn möglich, eine kartografische Darstellung **Deutschland**s aus einem **BTW**-Objekt dargestellt, wenn nicht wird eine Liste aller **Bundesländer** angezeigt.

zeigeInformationen(btw : BTW)
Erstellt, wenn möglich, die kartografische Darstellung.

faerbeBundeslaender(btw : BTW))
Färbt die einzelnen Bundesländer nach den Parteien, die die meisten Zweitstimmen haben. Diese Methode wird von zeigeInformationen benutzt.

GUISteuerung

Die **GUISteuerung** sorgt dafür, dass die aktuellste Version der **BTW** in einem **WahlFenster** visualisiert wird. Dafür bekommt sie das aktuellste **BTW**-Objekt der **Steuerung** und verteilt die Darstellungsarbeit an die Komponenten des **WahlFensters**.

aktualisiereWahlfenster()
Diese Methode startet eine komplette Aktualisierung eines **WahlFensters**. Hierbei werden auch alle Abhängigen Ansichten ebenfalls aktualisiert.

3.2 Datenhaltung

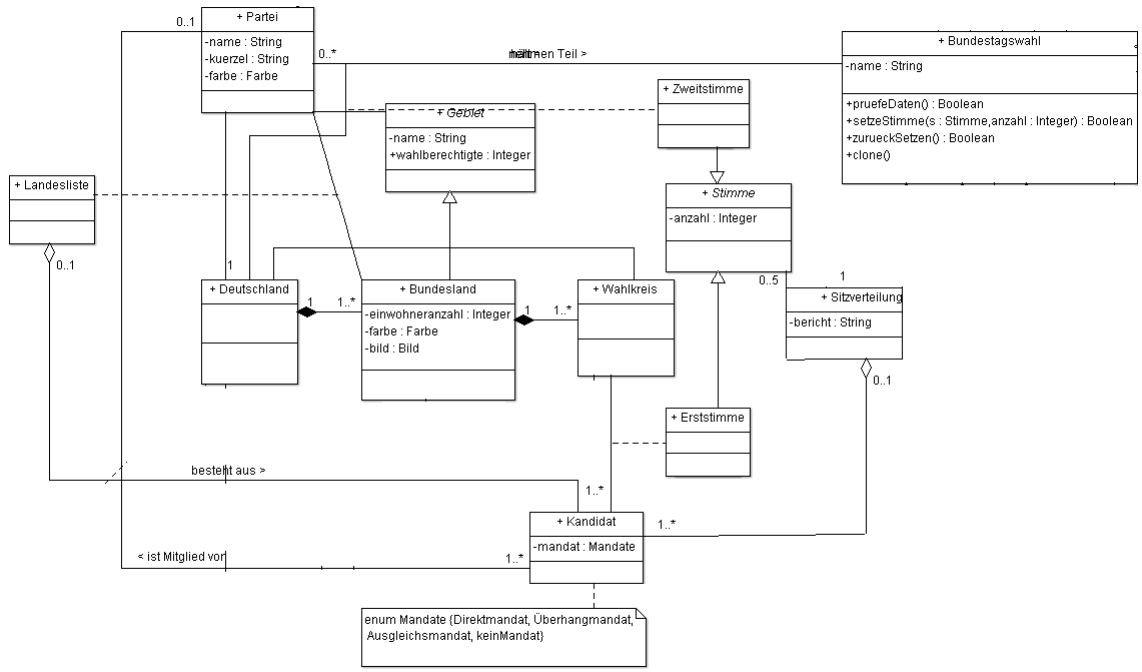

Abbildung 3: Datenhaltungs Komponente

Unter dem Datenmodell zählen alle Klassen, die Wahldaten oder die berechnete Sitzverteilung enthalten. Begründungen für Entwurfsentscheidungen und alle Klassen werden im folgenden aufgelistet:

`Bundestagswahl`

Eine Bundestagswahl stellt eine ganze Wahl dar. Es hat einen Namen wie "Bundestagswahl 2013". Dieser Name wird als Titel in der GUI auf dem betroffenen Tab angezeigt. Jede Bundestagswahl besitzt aufgrund der änderbaren Parteien pro Bundestagswahl eine eigene Liste an **`Partei`**-Objekten. Des weiteren gibt es ein **`Sitzverteilung`**-Objekt.
Folgende Methoden sind enthalten:

`prüfeDaten()`

Überprüft, ob die eingegebenen Stimmen gültig sind und unterzieht diese einem Konsistenz-Test. Die Methode wird nach jeder Stimmveränderung ausgeführt. Falls der Test nach einer Stimmveränderung fehlschlägt, wird die Stimme mithilfe der **`Chronik`** zurückgesetzt und eine Fehlermeldung ausgegeben.

`setzeStimme(s:Stimme,anzahl:Integer) : Boolean`

Wird von der GUI über die Steuerung aufgerufen. Es wird das zu verändernde **`Stimme`**-Objekt und die neue Anzahl der Stimmen übergeben. Es findet eine Vorüberprüfung statt, in der ermittelt wird was genau verändert wird. Es dürfen Erststimmen nur auf Wahlkreisebene und Zweitstimmen auf Wahlkreis-, Landes- und Bundesebene verändert werden. Die zu verändernde **`Stimme`**

wird geklont und in der **Chronik** gespeichert. Dabei können drei Fälle auftreten:
Falls die assoziierte Klasse von **Stimme**:
... ein **Wahlkreis**-Objekt ist, wird der enthaltene Wert "anzahl" verändert in den Wert des übergebenen Parameters "anzahl".
... ein **Bundesland**-Objekt ist, wird die Differenz der anzahl an Stimmen auf alle Wahlkreise des betroffenen Bundeslandes iterativ hinzugefügt oder abgezogen.
... ein **Deutschland**-Objekt ist, wird die Differenz der Anzahl an Stimmen auf alle Wahlkreise der Bundesländer iterativ hinzugefügt oder abgezogen.
Zum Schluss wird die Methode *prüfeDaten()* aufgerufen.

zurueckSetzen() : Boolean
Ruft die Methode *restauriereStimme()* in der **Chronik** auf und setzt eine **Stimme** zurück. Es wird dabei die Methode *setzeStimme(s:Stimme, anzahl:Integer)* verwendet um die Stimmen zu ändern. Falls das Rücksetzen fehlschlägt ist der Rückgabewert "false". Andernfalls wird erneut *restauriereStimme()* aufgerufen um den neu erstellten Chronik-Eintrag in*setzeStimme(s:Stimme, anzahl:Integer)* zu entfernen. Anschließend wird "true" zurückgegeben.

clone() : Bundestagswahl
Macht eine "deep copy" von der aktuellen Bundestagswahl und gibt dies zurück.

Partei
Ein Objekt dieser Klasse spiegelt eine vertretene Partei des Bundestages wieder. Jede Partei besitzt einen Namen, ein Kürzel und eine Farbe. Die Farbe wird verwendet, um in der kartografischen Ansicht die Bundesländer mit der Partei einzufärben, die in diesem Bundesland die meisten Zweitstimmen hat.

Gebiet
Eine Abstrakte Klasse, die ein Gebiet darstellt. Es erbt den Klassen **Deutschland**, **Bundesland** und **Wahlkreis**. Gebiet besitzt eine Assoziationsklasse (**Zweitstimme**) mit **Partei**. Jedes Gebiet besitzt einen Namen und ein Wert mit der Anzahl der Wahlberechtigten.

Deutschland
Deutschland besitzt eine Liste an **Bundesland**-Objekten, die in der Wahl benutzt wurden. Diese Klasse wird beispielsweise verwendet, wenn die Zweitstimmen einer Partei in der **Bundesansicht** verändert wurde.

Bundesland
Bundeland besitzt eine Liste an **Wahlkreis**-Objekten, die in der Wahl benutzt wurden. Diese Klasse wird beispielsweise verwendet, wenn die Zweitstimmen einer Partei in der **Landesansicht** verändert wurde. Diese Klasse enthält zusätzlich einige weitere Attribute. Diese sind Einwohnerzahl (zur Berechnung der möglichen Anzahl der Sitze pro Bundesland), sitze (berechnete Anzahl der möglichen Sitze),

Farbe (Farbe der Partei mit den meisten Zweitstimmen, wird im Mandatsrechner ausgemacht) und Bild um das Wappen des Bundeslandes zu Speichern. Der Wappen wird in der Landesansicht angezeigt.

Wahlkreis
Ein **Wahlkreis** besitzt zusammen mit **Kandidat** eine Assotiationsklasse **Erststimme**, was die Erststimme pro Wahlkreis und Kandidat/Partei wiederspiegelt. In der **Wahlkreisansicht** werden Objekte dieser Klasse von einem spezifischen Bundesland dargestellt.

Stimme
Dies ist eine Abstrakte Klasse, die eine Oberklasse von **Erststimme** und **Zweitstimme** ist. Diese Klasse wird bei Stimmveränderungen verwendet um ausmachen zu können, welche Stimme genau verändert wurde.

Erststimme
Darf nur als Assoziationsklasse zwischen **Wahlkreis** und ein **Kandidat** existieren. Dies stellt sicher, dass die Erststimme einzig auf Wahlkreisebene verändert werden kann/darf. Der Grund für diese Entscheidung ist, die Komplexität bei Veränderung der Erststimmen auf Bundes- und Landesebene zu vermeiden.

Zweitstimme
Eine Assoziationsklasse zwischen **Partei** und **Gebiet**. Ermöglicht die Veränderung der Zweitstimmen auf Bundes- Landes- als auch Wahlkreisebene.

Kandidat
Ein Kandidat kann entweder Direktmandat, Überhandmandat, Ausgleichsmandat oder überhaupt kein Mandat sein. Des weiteren kann ein Kandidat Abgeordneter sein, falls dieser ein Attribut einer **Sitzverteilung** besitzt.

Landesliste
Die Assoziationsklasse zwischen **Partei** und **Bundesland**. Beinhaltet eine Liste an Kandidaten. Die Reihenfolge der Kandidaten ist hierbei entscheidend, da dies entscheidet welche Kandidaten als erstes ein Sitz zugeordnet wird.

Sitzverteilung
In dieser Klasse wird ein Bericht angelegt, dass zeigen soll, welcher Sitz wie entstanden, und welcher Partei zugeordnet ist. Es enthält eine Liste an Kandidaten. Die Informationen werden mit dem **Mandatsrechner** generiert. Es wird hierbei zwischen Direktmandaten, Überhangmandaten und Ausgleichsmandaten unterschieden. Die Informationen werden in **BerichtsFenster** visualisiert.

3.3 Steuerung

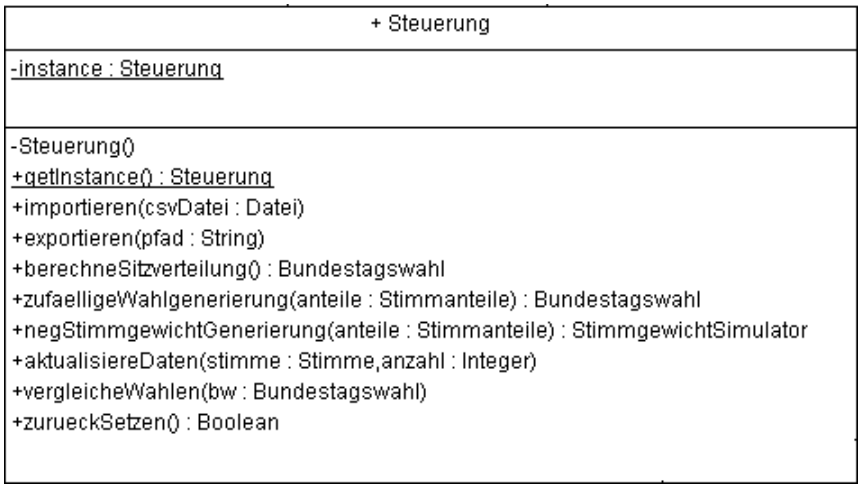

Abbildung 4: Steuerung Komponente

Die **Steuerung** verbindet die unterschiedlichen Kompententen (GUI, Datenmodelle, etc.). Es wurde deshalb das Entwurfsmuster Fassade für den Entwurf der Klasse genommen. Da es nur ein eindeutiges Objekt von der Klasse geben soll, wird zusätzlich Einzelstück als ein weiteres Entwurfsmuster genutzt. Die Steuerung hält für jeden Tab im **Programmfenster** ein **Buntestagswahl**-Objekt.

Steuerung()
: Der private Konstruktor wird für das Entwurfsmuster Einzelstück gebraucht. Dadurch ist es nicht möglich eine neue Instanz von der Klasse zu erstellen.

getInstance() : Steuerung
: Diese Methode gehört ebenfalls zum Entwurfsmuster Einzelstück, da das private Attribut nur als Rückgabewert einer Methode übergeben werden kann.

importieren(csvDatei : Datei)
: Die in der GUI ausgewählte Datei wird an den **ImportExportManager** gesendet. Wenn die Formatierung und der Inhalt der Datei stimmen, wird das übergebene **Bundestagswahl**-Objekt gespeichert.

exportieren(pfad : String)
: Es wird ein **Bundestagswahl**-Objekt und ein Dateipfad an den **ImportExportManager** gesendet, damit die Klasse eine CSV-Datei erstellt und diese speichert.

berechneSitzverteilung() : Bundestagswahl
: Das aktuelle **Bundestagswahl**-Objekt wird im passenden Mandatsrechner ausgewertet. Das Ergebnis wird als Rückgabewert von der **Steuerung** zurückgegeben.

zufälligeWahlgenerierung(anteile : Stimmanteile) : Bundestagswahl
: Es wird mit Hilfe des **Wahlgenerators** und der **Stimmenanteile** eine **Bundestagswahl** erstellt und zurückgegeben.

negStimmgewichtGenerierung(anteile : Stimmanteile)
Diese Methode gibt einen **NegStimmgewichtSimulator** zurück, der das negative Stimmgewicht zweier, miteinander verwandter Bundestagswahlen zum Ausdruck bringt.

aktualisiereDaten(stimme : Stimme, anzahl : Integer)
Die Methode speichert den alten Wert in der Chronik und kopiert den neuen Wert über Deep copy in die Daten.

vergleicheWahlen(btw1 : Bundestagswahl, btw2 : Bundestagswahl)
Es werden zwei Wahlen an das **Wahlvergleich**-Objekt gesendet.

zurueckSetzen() : Boolean
Der aktuellen Stand des Programmes wird mit dem alten Stand ersetzt. Bei einer erfolgreichen Zurücksetzten ist der Rückgabewert der Methode true;

3.4 Import/ Export

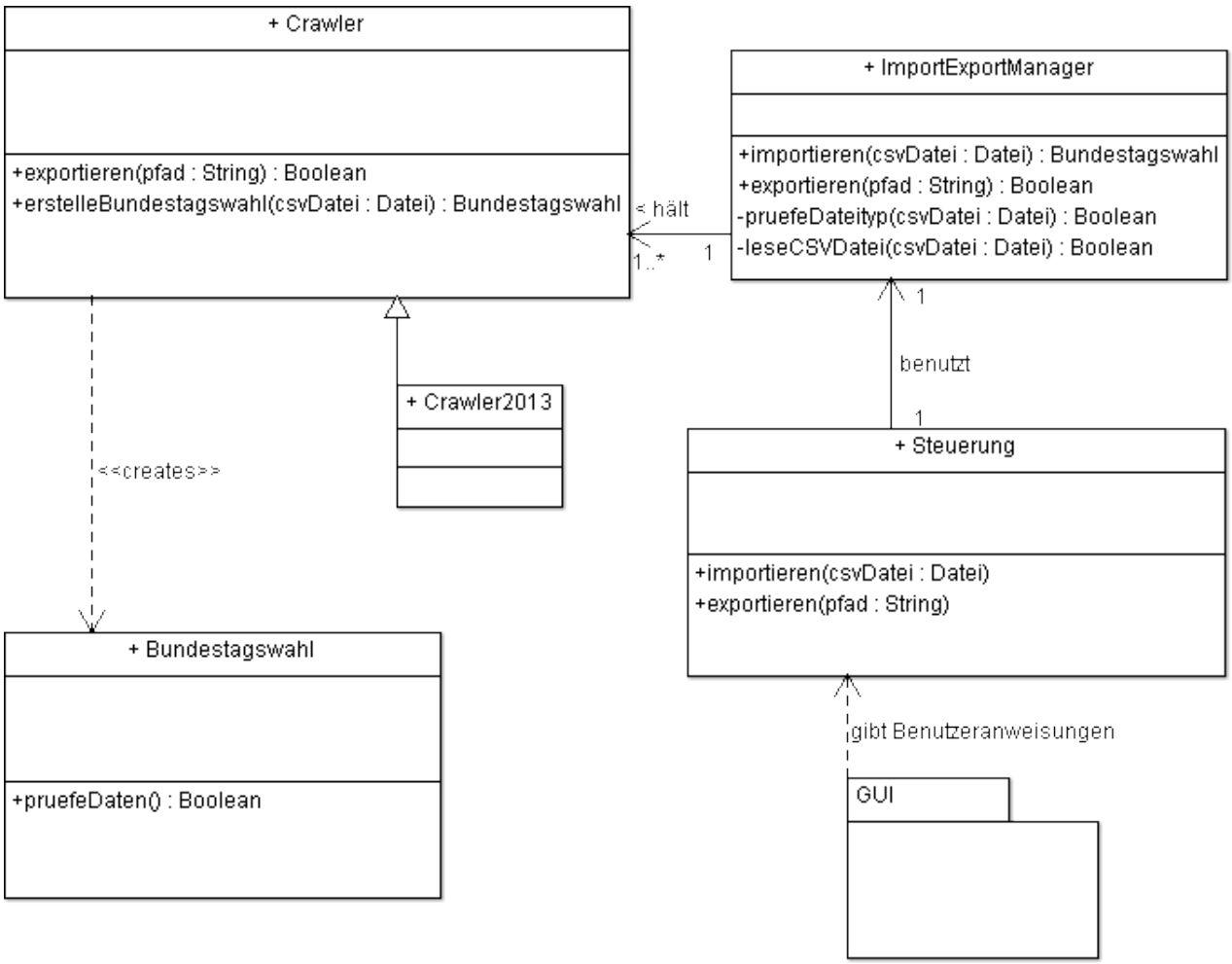

Abbildung 5: Import/Export Komponente

Hier sieht man den Aufbau des Import- bzw. Exportmoduls. Zur Übersichtlichkeit werden die zum Importieren/Exportieren nicht notwendigen Methoden in **`Steuerung`** und die genaue Struktur hinter **`Bundestagswahl`** und der GUI ausgeblendet.
Mit dem Programm wird nur ein vorimplementierter Crawler mitgegeben, der .csv-Dateien, die dem Format der .csv-Datei zur Bundestagswahl 2013 der Bundeswahlleiter-Webseite entsprechen, auswerten kann - dies ist **`Crawler2013`**. Um jedoch die Möglichkeit zu garantieren, nachträglich weitere Crawler hinzuzufügen, haben wir uns dafür entschieden, eine abstrakte Oberklasse **`Crawler`** zu verwenden, von der **`Crawler2013`** erbt, und alle vorhandenen Crawler von der Klasse **`ImportExportManager`** halten zu lassen.
Ausgelöst wird der ganze Import- bzw. Exportvorgang durch eine Benutzerinteraktion (z.B. Betätigen des Laden- Knopfs im Menü), worauf **`Steuerung`** die entsprechenden Methoden von **`ImportExportManager`** ausführt.

ImportExportManager

importieren(csvDatei : Datei) : Bundestagswahl
Diese öffentliche Methode führt zuerst die private Methode *pruefeDateityp()* aus. Wenn diese true zurückgibt, wird die ebenfalls private Methode *leseCSVDatei()* ausgeführt. Wurde nun eine gültige Bundestagswahl zurückgegeben, wird diese an die Steuerung zurückgegeben, andernfalls ein Fehler ausgegeben.

pruefeDateityp(csvDatei : Datei) : Boolean
Prüft, ob es sich bei der gegebenen Datei um eine .csv-Datei handelt. Wenn dies der Fall ist, wird true zurückgegeben, andernfalls false.

leseCSVDatei(csvDatei : String) : Boolean
Durchläuft die Crawler-Liste und lässt die darin enthaltenen Crawler nacheinander versuchen, die .csv-Datei auszuwerten und mit den gewonnenen Informationen eine Bundestagswahl zu erstellen und zu füllen. Gelingt es einem Crawler, wird der Durchlauf abgebrochen und true zurückgegeben.

Crawler

exportieren(Pfad : String) : Boolean
Nimmt die aktuelle Bundestagswahl der Steuerung und schreibt die zugehörigen Wahldaten, ohne die berechneten Daten, in eine dem Pfad entsprechende .csv-Datei, deren Format Crawler bestimmt wird. Bei einem auftretenden Fehler wird false zurückgegeben.

erstelleBundestagswahl(csvDatei : Datei) : Bundestagswahl
Erstellt anhand einer .csv-Datei eine neue Bundestagswahl und versucht sie zu füllen. Wenn nicht alle nötigen Daten vorhanden sind, oder die Struktur der .csv-Datei nicht der vom Crawler geforderten Struktur entspricht, wird ein Fehler ausgegeben und der Import abgebrochen.

3.5 Wahlgenerierung

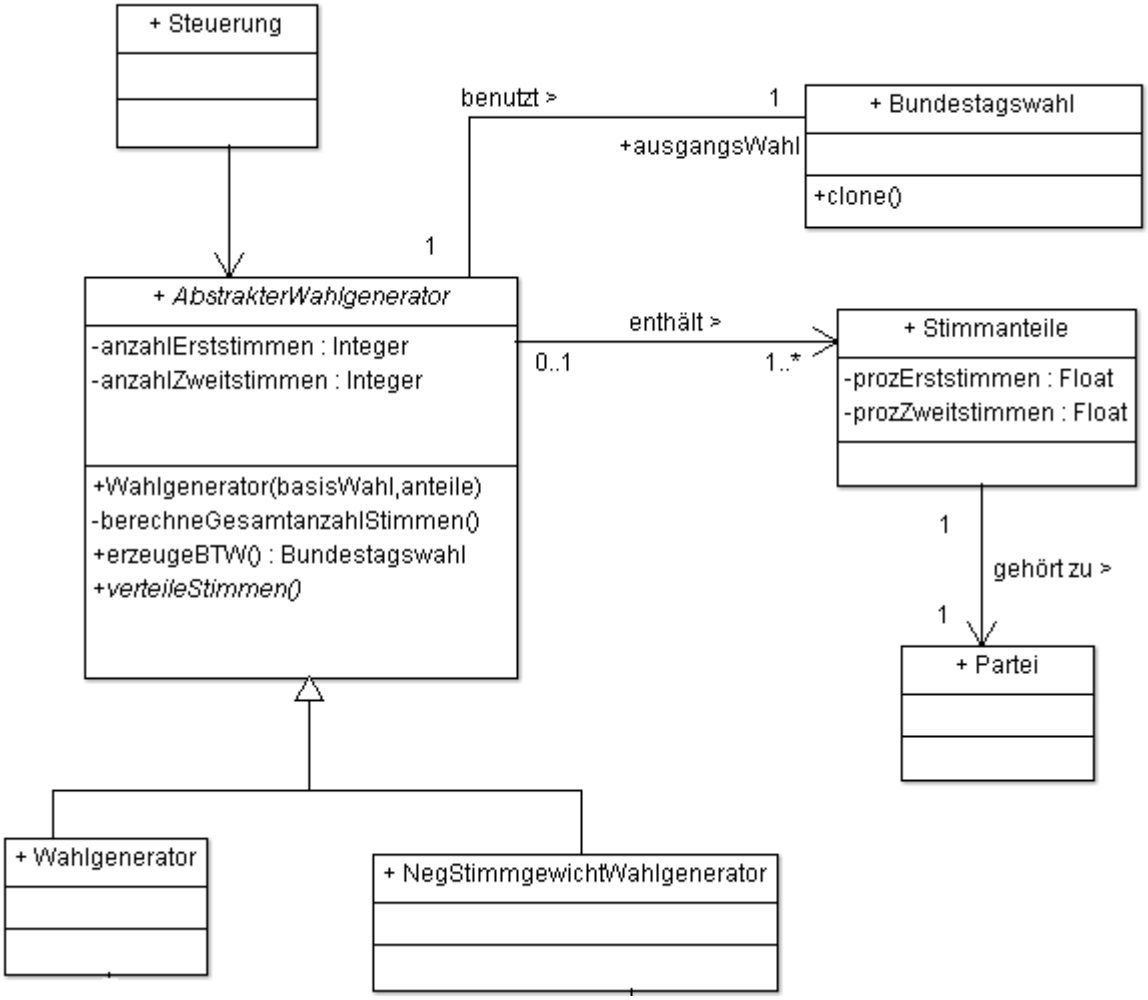

Abbildung 6: Wahlgenerierungs-Komponente

Zur Übersichtlichkeit werden in den Klassen **`Partei`**, **`Steuerung`** und **`Bundestagswahl`** nur die für dieses Modul relevanten Informationen angezeigt.

Mit diesem Modul können **`Bundestagswahl`** Objekte anhand vorher definierter **`Stimmanteile`** auf Bundesebene generiert werden. Bei der Klasse **`Stimmanteile`** handelt es sich um eine Liste aller Parteien mit prozentualen Anteilen der Erst- und Zweitstimmen auf Bundesebene. Des weiteren benötigt der Wahlgenerator eine **`basisWahl Bundestagswahl`** um Daten wie beispielsweise **`Bundesländer`**, **`Wahlkreise`** und **`Wahlberechtigte`** zur Verfügung zu haben. Aus dieser basisWahl wird eine tiefe Kopie erstellt, deren Stimmzahlen anschließend verändert werden.

Neben dem `Wahlgenerator`, der alle Stimmen der jeweiligen Parteien zufällig auf Wahlkreise verteilt gibt es noch den `NegStimmgewichtWahlgenerator`. Dieser er-

zeugt Bundestagswahlen, die die Voraussetzungen erfüllen, welche für die Simulation des negativen Stimmgewichts benötigt werden.

Dabei muss bei mindestens einer Partei der prozentuale Anteil ihrer relevanten Zweitstimmen größer als der prozentuale Anteil ihrer Mandate sein. Relevante Zweitstimmen sind all diejenigen Zweitstimmen, die auf Landeslisten abgegeben werden, die keine Überhangmandate erzielen.

Wahlgenerator(basisWahl : Bundestagswahl, anteile : Stimmanteile)
Der Konstruktor dieser Klasse. Wird verwendet um einen neuen *Wahlgenerator* zu erstellen. Hier werden die Attribute basisWahl, anteile, erststimmenAnzahl und zweitstimmenAnzahl gesetzt.

berechneGesamtanzahlStimmen()
Diese Methode ist privat und wird von dem Konstruktor verwendet um die Attribute *anzahlErststimmen* und *anzahlZweitstimmen* zu berechen. Hierzu werden die Stimmanteile mithilfe der Anzahl aller Wahlberechtigten in absolute Zahlen für Erst- und Zweitstimmen umgerechnet.

erzeugeBTW() : Bundestagswahl
Erzeugt eine neue Bundestagswahl auf der Grundlage der basisWahl und füllt diese mit den Erst- und Zweitstimmen.

verteileStimmen()
Diese Methode verteilt alle Erst- und Zweitstimmen auf die Wahlkreise der Bundestagswahl. Diese Methode muss in jeder Unterklasse von Wahlgenerator implementiert werden.

3.6 Negatives Stimmgewicht simulieren

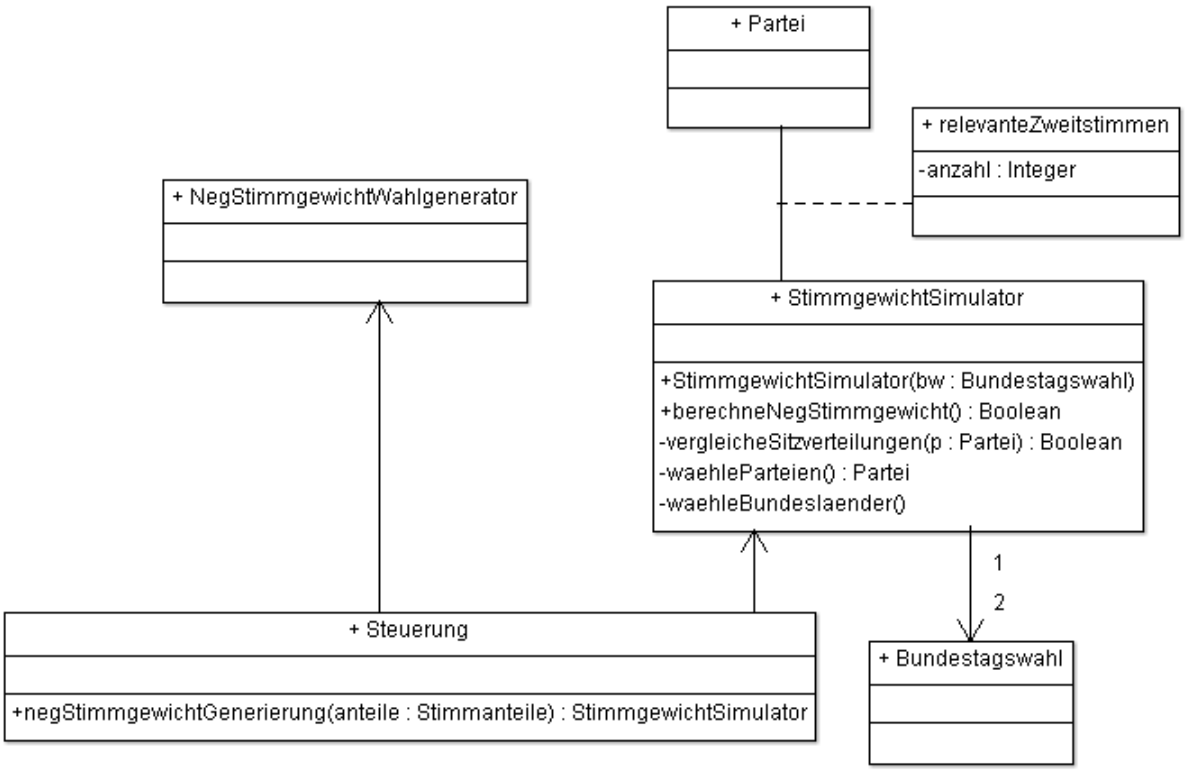

Abbildung 7: Negatives Stimmgewicht Komponente

Hier sieht man den Aufbau der Komponente, die für das Simulieren des negativen Stimmeffekts zuständig ist. Zur Übersichtlichkeit werden die nicht notwendigen Methoden und Klassen ausgeblendet.
Zu einer von **NegStimmgewichtWahlgenerator** erzeugten **Bundestagswahl**, soll durch Simulation eine verwandte **Bundestagswahl** in der Art erzeugt werden, so dass zwischen diesen zwei Bundestagswahlen der Effekt des negativen Stimmgewichts auftritt.

StimmgewichtSimulator(btw : Bundestagswahl)
: Das ist der Konstruktor der Klasse. Dieser wird verwendet, um einen neuen **StimmgewichtSimulator** zu erstellen. Hier wird von der als Parameter übergebenen **Bundestagswahl** die Sitzverteilung berechnet. Zudem wird von der **Bundestagswahl** eine "deep copyërstellt und als Attribut abgespeichert. Danach werden die relevanten Zweitstimmen (siehe Wahlgenerierung) aller Parteien berechnet.

berechneNegStimmgewicht() : Boolean
: Ruft die privaten Methoden *waehleParteien() : Partei* und *waehleBundeslaender() : Bundesland* auf. Dann werden die Zweitstimmen der ersten **Partei** aus der erhaltenen Liste in einem geeigneten **Bundeesland**

schrittweise um einen konstanten Faktor erhöht. Dies geschieht maximal solange bis sie nicht mehr erhöht werden können, weil sie größer als die Anzahl der Wahlberechtigten sind. Falls ein negatives Stimmgewicht im Vergleich zu der ursprünglichen Bundestagswahl ermittelt wurde, bricht die Suche ab und es wird true zurückgegeben.

vergleicheSitzverteilungen(p : Partei) : Boolean Vergleicht die Sitzverteilungen der beiden in Stimmgewicht gegebenen Bundestagswahlen. Wenn die Sitzanzahl für die gegebene Partei in der Ursprungsbundestagswahl größer ist als in der neu berechneten Bundestagswahl, tritt negativer Stimmeffekt auf und es wird false ausgegeben, andernfalls true.

waehleParteien() : Partei In dieser privaten Methode werden die Parteien gewählt, für die negatives Stimmgewicht auftreten kann (Bedingung: siehe Wahlgenerierung).

waehleBundeslaender() : Bundesland In dieser privaten Methode werden die Bundesländer, abhängig von den Parteien, gewählt, für die negatives Stimmgewicht auftreten kann (Bedingung: siehe Wahlgenerierung).

3.7 Chronik

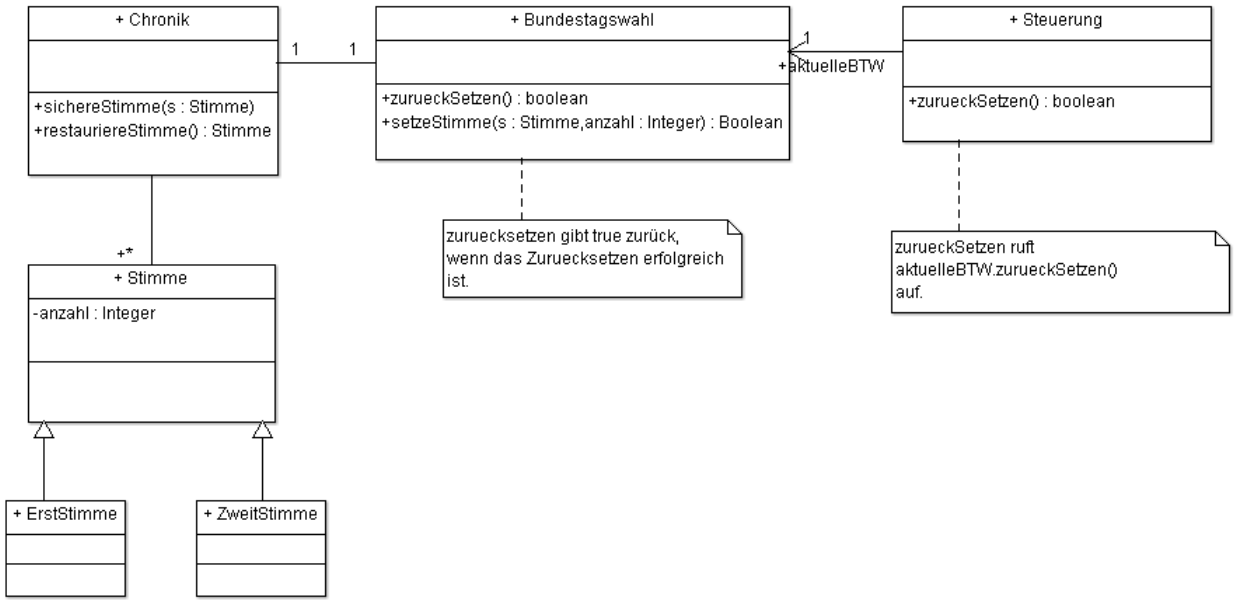

Abbildung 8: Chronik-Komponente

Die Klasse **Chronik** gibt dem Programm die Funktionalität, Veränderungen an den Stimmen rückgängig zu machen. Jede **Bundestagswahl** hat hierbei eine eigene Chronik. **Chronik** wird in dem Konstruktor von **Bundestagswahl** erzeugt, und ist daher in jedem **Bundestagswahl**-Objekt enthalten. Es besitzt eine Menge von **Stimmen**-Objekten. Bei jeder Veränderung wird ein neues **Stimmen**-Objekt angelegt, was die Veränderung wiederspiegelt. Die Methode **sichereStimme** wird von dem **Bundestagswahl**-Objekt bei jedem Aufruf von **setzeStimme** aufgerufen.

Methoden

sichereStimme(s : Stimme)
: Diese Funktion wird von dem assoziierten **Bundestagswahl**-Objekt innerhalb der *setzeStimme*-Funktion aufgerufen. Falls bereits fünf **Stimmen**-Objekte vorhanden sind, wird das älteste entfernt.

restauriereStimme() : Stimme
: Wird von der **Steuerung** über die aktuelle Bundestagswahl mit der Funktion *zurueckSetzen()* aufgerufen und gibt die zuletzt hinzugefügte **Stimme** zurück. Die Bundestagswahl ersetzt dann die aktuelle Stimme mit der restaurierten Stimme.

3.8 Mandatsrechner

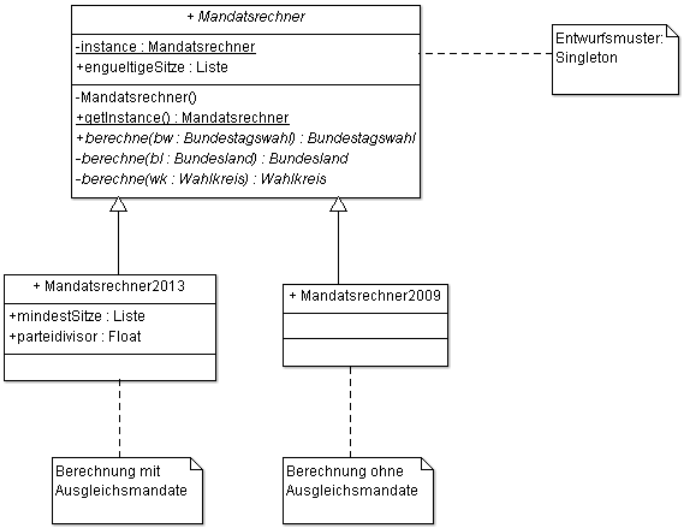

Abbildung 9: Mandatsrechner-Komponente

Die Berechnung der Wahl wird mit Hilfe des **`Mandatsrechners`** realisiert. Es stehen die Klasse
`Mandatsrechner2013`, die das Berechnungsverfahren von der Bundestagswahl 2013 benutzt und die Klasse **`Mandatsrechner2009`**, die das Berechnungsverfahren von der Bundestagswahl 2009 benutzt zur Verfügung. Beide Klassen erben von der abstrakten Klasse **`Mandatsrechner`**. Dadurch besteht die Möglichkeit, weitere Berechnungsverfahren in späteren Versionen zu dem Programm hinzuzufügen. Da nur ein Objekt von dem **`Mandatsrechner`** gebraucht wird, wird das Entwurfsmuster Einzelstück eingesetzt. Deswegen hält die Klasse einen privaten Konstruktor. Die abstrakten Methoden *`berechne()`* werden überladen, damit sie durch ihre Eingabeparameter spezifiziert werden. Diese werden dann in den Unterklassen je nach Wahlgesetz angepasst. Neben der Berechnung wird ein Bericht über die Sitzverteilung erstellt, der zum Nachvollziehen der Sitzverteilung helfen soll.

`berechne(wk: Wahlkreis):Wahlkreis`
: Es werden die Stimmen aus den jeweiligen Wahlkreis ausgewertet. Dabei wird der Wahlkreissieger bestimmt und die Anzahl der Zweitstimme von jeder Partei. Die Auswertung wird danach wieder in das Wahlkreis-Objekt geschrieben.

`berechne(bl: Bundesland):Bundesland`
: Um das Bundesland zu berechnen, müssen vorher alle Wahlkreise berechnet werden. Deswegen werden alle Wahlkreise, die ein Bundesland hält, neu berechnet. Die Berechnung der einzelnen Bundesländer erfolgt parallel. Nachdem die berechneten Wahlkreise im Bundesland gespeichert wurden, wird das Bundesland berechnet. Hier wird das Verhältnis der Parteien im Bundesland berechnet, damit später klar ist wie viele Sitze eine Partei in diesem Bundesland bekommt. Diese Ergeb-

nisse werden, wie beim Wahlkreis, im Bundeslandobjekt gespeichert und danach zurückgegeben.

berechne(btw: Bundestagswahl):Bundestagswahl
Diese öffentliche Methode berechnet zuerst alle Bundesländer die sich in der Klasse befinden. Nachdem alle Bundesländer erfolgreich berechnet wurden, wird die endgültige Sitzverteilung nach dem jeweiligen Wahlgesetz berechnet. Die Sitzverteilung wird dann in dem Bundestagswahlobjekt gespeichert. Das Bundestagswahlobjekt wird danach wieder an die Steuerung zurückgegeben.

erstelleBericht(Zeile : String)
Während der Berechnung wird nebenbei eine Sitzverteilungsbericht verfasst, der beschreiben soll, wie die Sitzverteilung entstanden ist. Dabei wird die Methode immer aufgerufen, wenn eine Partei einen Sitz in der Sitzverteilung bekommen hat. Dies wird dann mit einer Zeile im Bericht protokolliert.

3.9 Wahlvergleich

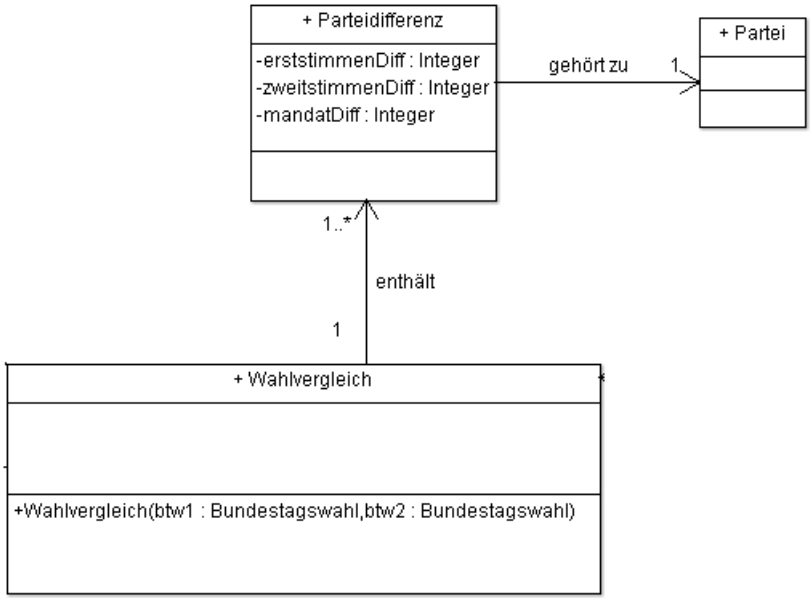

Abbildung 10: Wahlvergleich-Modul

Dieses Modul ermöglicht den Vergleich von zwei Bundestagswahlen. Die Klasse **`Wahlvergleich`** enthält zwei Bundestagswahlen sowie eine Liste aller relevanten Parteien mit den Differenzen von Erst-, Zweitstimmen sowie der Anzahl von Mandaten.

3.10 Meldung

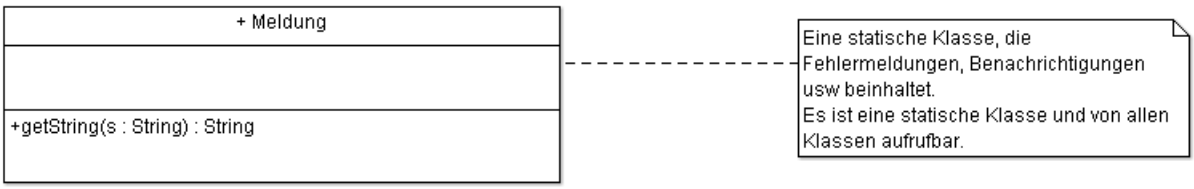

Abbildung 11: Meldungs-Klasse

Die Klasse **`Meldung`** ist verantwortlich für Fehlermeldungen, Benachrichtigungen und Fenstertexte. Es ist eine statische Klasse. Die Funktion *`getString`* gibt zu einem gegebenen Schlüssel ein String zurück. Die Strings dieser Klasse werden in einem externen Textdokument gelagert.

4 Sequenzdiagramme

4.1 Berechnung der Sitzverteilung

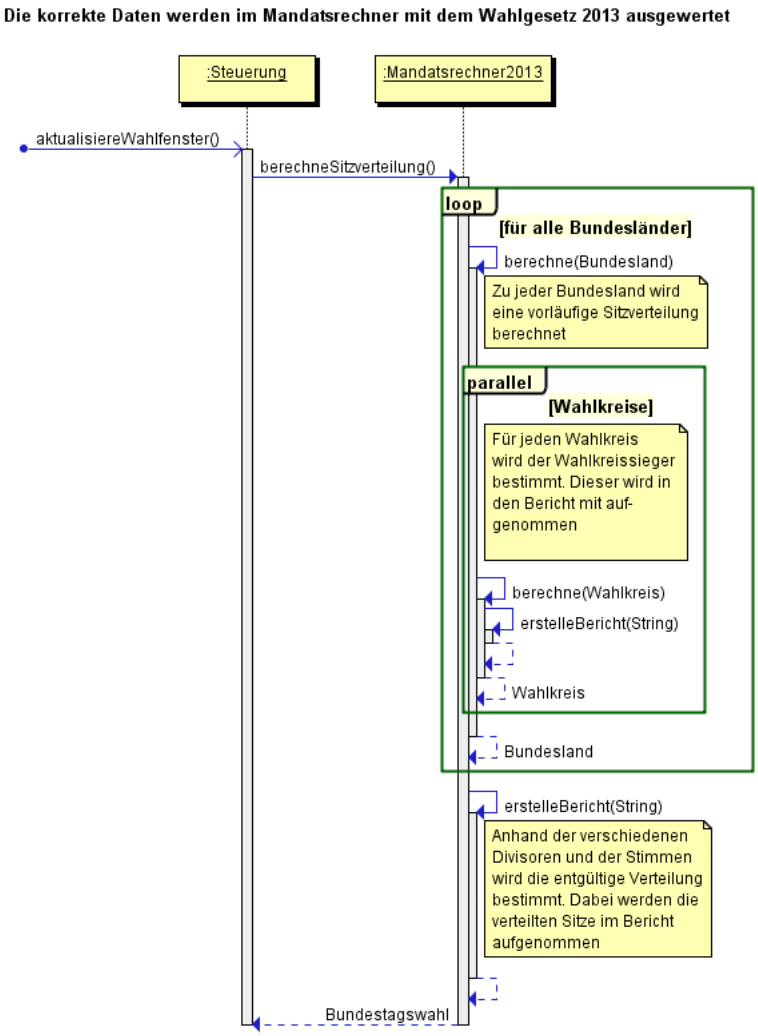

Abbildung 12: Sequenzdiagramm zur Stimmenänderung

Das **`Bundestagswahl`**-Objekt wir mit dem **`Mandatsrechner2013`** ausgewertet.

4.2 DeepCopy

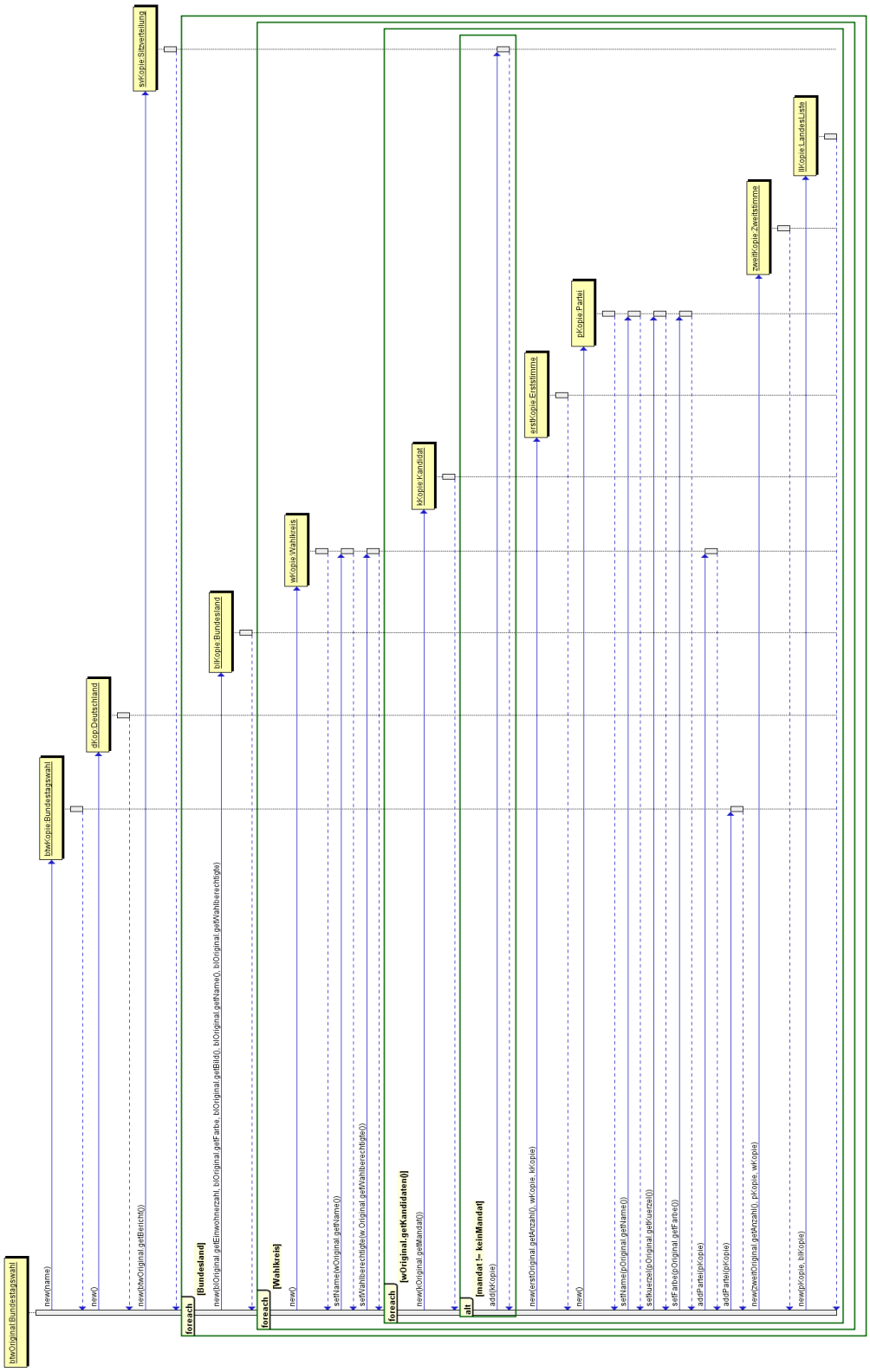

Abbildung 13: Kopieren einer Bundestagswahl

Dieses Sequenzdiagramm zeigt, wie man ein gesamtes **Bundestagswahl**-Objekt mit Hilfe der clone()-Methode kopiert.
Es beginnt mit der Erstellung eines neuen **Bundestagwahl**-Objekts. In dieses wird ein **Sitzverteilung**s- und **Deutschland**-Objekt generiert, welches alle **Bundesländer** des Originals neu in sich anlegt. Für jedes **Bundesland** müssen die originalen **Wahlkreise** kopiert werden. Die **Wahlkreis**e haben jeweils eine Liste von **Kandidat**en, wodurch durch die Assoziation zwischen **Wahlkreis** und **Kandidat** noch die **Zweitstimme**n neu generiert werden. **Kandidat**en werden, wenn sie ein Mandat haben, zur neuen **Sitzverteilung** hinzugefügt. Jeder **Kandidat** ist Mitglied in einer **Partei**, die wiederum kopiert werden muss. Aus dieser und dem dazugehörigen **Gebiet** entsteht die neue **Zweitstimme** und durch aus der **Partei** und dem **Bundesland** entsteht die **Landesliste**, was die Kopierung beendet.

4.3 Import

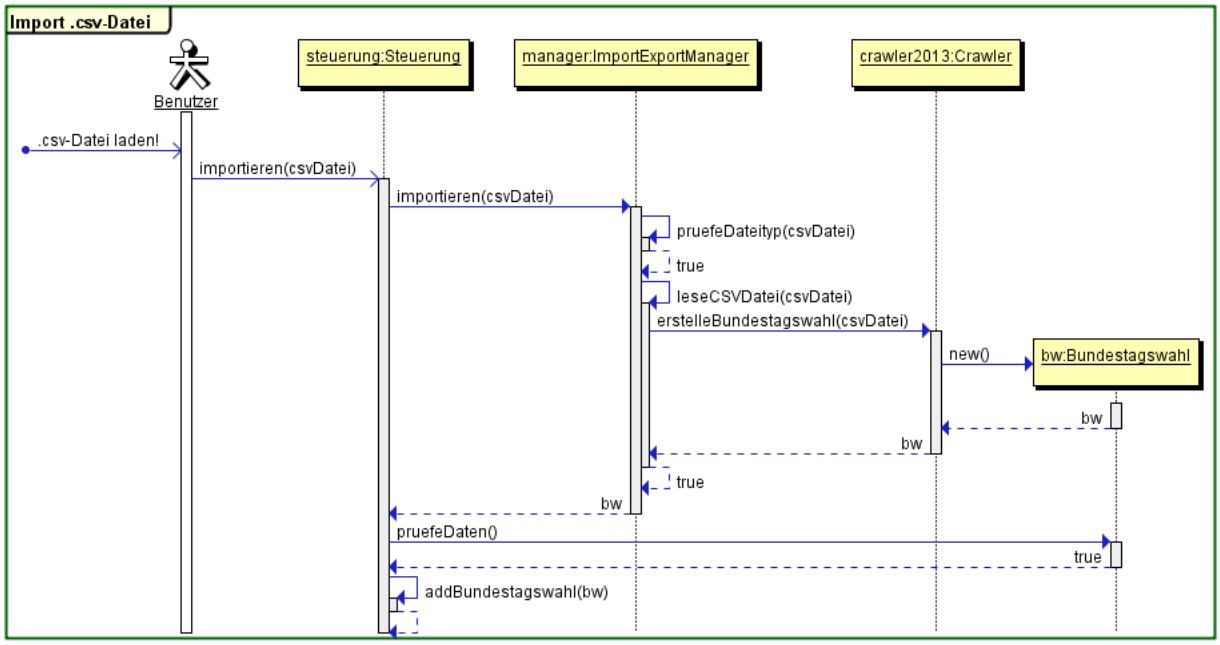

Abbildung 14: Import Sequenzdiagramm

Hier wird eine gültige .csv-Datei, d.h. Format und Inhalt betreffend, importiert. Vorher wurde bereits ein Datei-Objekt csvDatei erstellt, das nun übergeben wird.

4.4 Wahlgenerierung

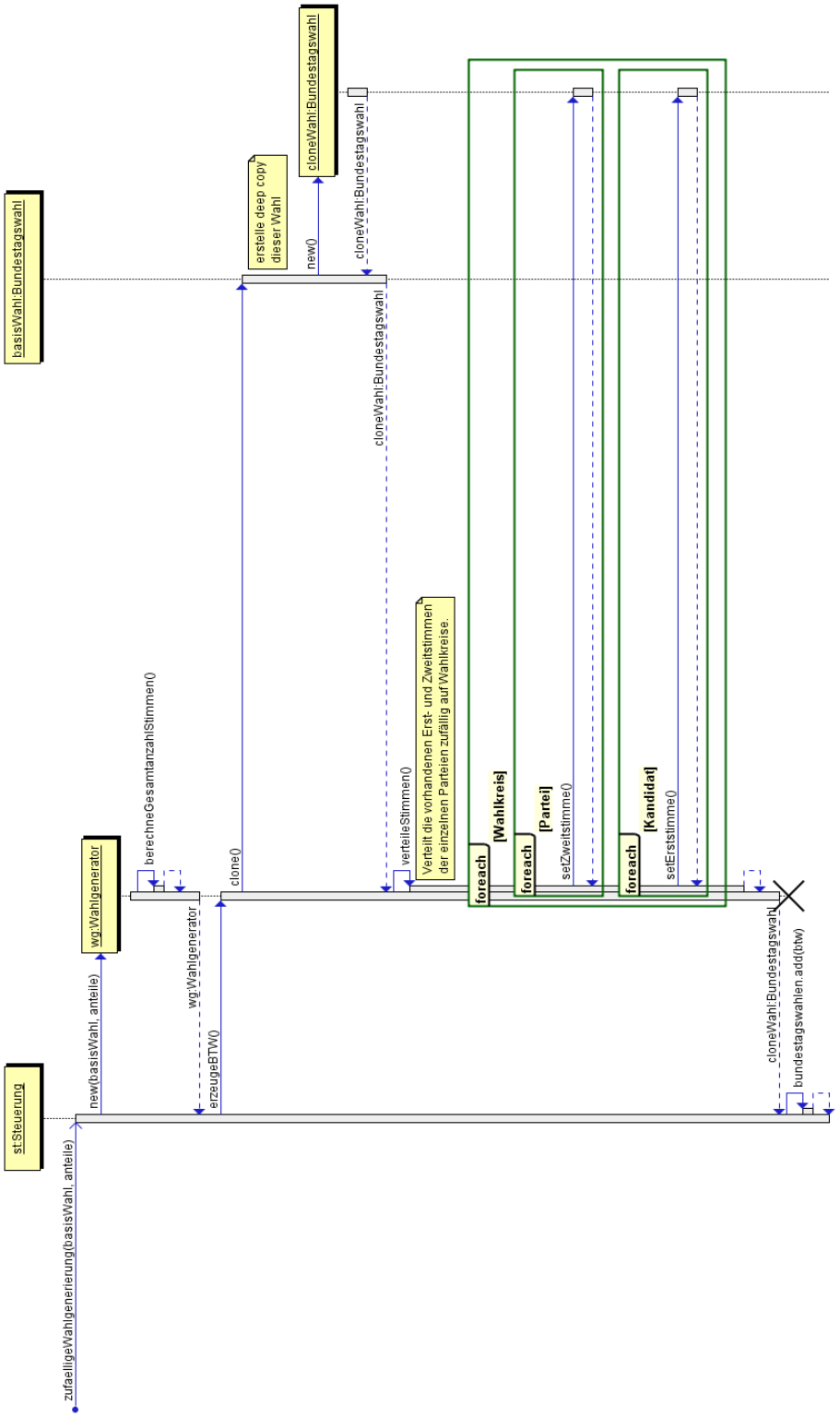

Abbildung 15: Sequenzdiagramm zur Wahlgenerierung

In der Methode *zufaelligeWahlgenerierung(basisWahl : Bundestagswahl, anteile : Stimmanteile) : Bundestagswahl* wird zuerst ein neuer Wahlgenerator erzeugt. In dem Konstruktor des Wahlgenerators werden die absoluten Stimmzahlen berechnet und alle Attribute gesetzt.
Anschließend wird von der Steuerung die Methode *erzeugeBTW()* ausgeführt. In dieser Methode wird zuerst eine tiefe Kopie der basisWahl erstellt. In dieser Kopie werden dann die Stimmen auf alle Wahlkreise verteilt. Die Erststimmen auf die Kandidaten und die Zweitstimmen auf die Parteien. Diese Wahl wird am Ende als Ergebnis der Methode berechneBTW() zurückgegeben und in der Steuerung in die Liste alle Bundestagswahlen eingefügt.

4.5 Negatives Stimmgewicht

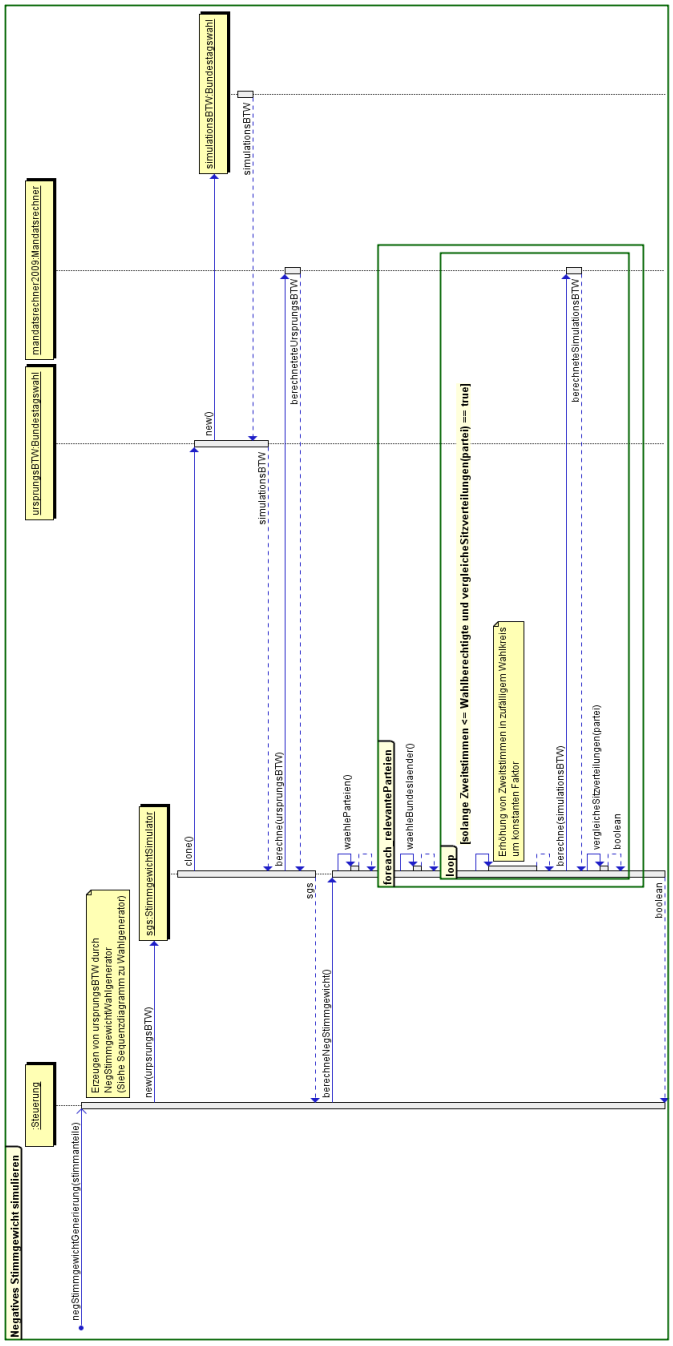

Abbildung 16: Sequenzdiagramm zum negativen Stimmgewicht

In diesem Sequenzdiagramm wird der Fokus daraufgelegt, wie zu der vom **`NegStimmgewichtWahlgenerator`** erzeugten **`Bundestagswahl`** ein **`Stimmgewichtsimulator`**-Objekt erzeugt und darauf dann mithilfe der Methode *`berechneNegStimmgewicht()`* eine, zur gegebenen Bundestagswahl verwandte, Wahl findet, so dass zwischen diesen der Effekt des negativen Stimmgewichts auftritt.

4.6 Chronik

4.6.1 Veränderung an den Stimmen

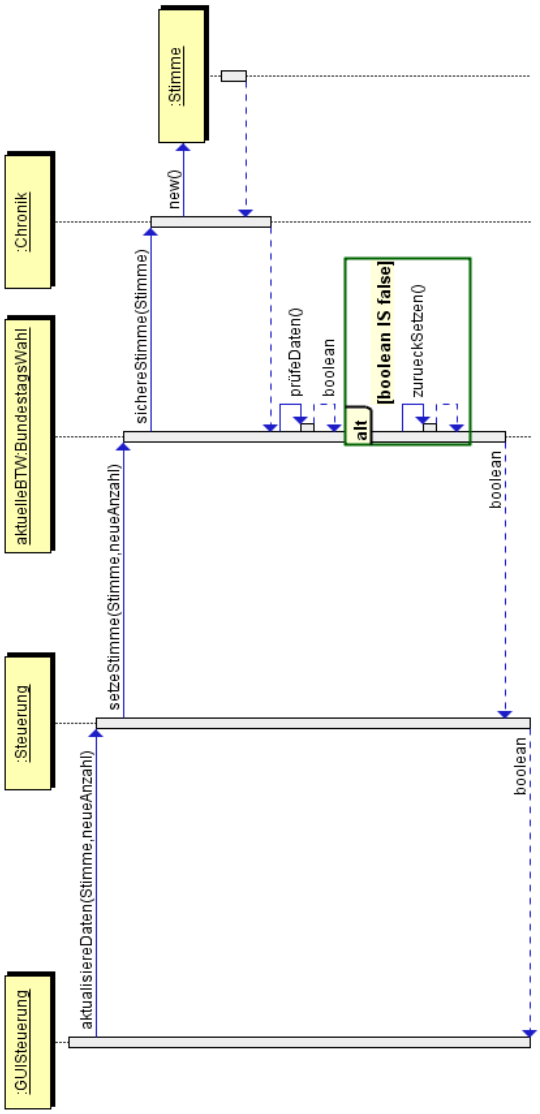

Abbildung 17: Sequenzdiagramm zur Sicherung des Stimmenobjektes

Beim Verändern einer Stimme wird von der GUI die Funktion `aktualisiereDaten(s:Stimme, neueAnzahl:Integer)` aufgerufen. Dies ruft in der aktuellen Bundestagswahl die Funktion `setzeStimme(s:Stimme, neueAnzahl:Integer)` auf. Hier wird nun zum Speichern der alten Stimme, das alte Stimmen-Objekt geklont und in der Chronik abgelegt. Anschließend wird die alte Stimme mit der neuen Stimme ersetzt (hier nicht dargestellt) und die Funktion `prüfeDaten(): Boolean` überprüft ob die Bundestagswahl mit der veränderten Stimme noch gültig ist. Im Falle einer ungültigen Bundestagswahl wird dies mithilfe der Chronik zurückgesetzt und “‘false” zurückgegeben.

4.6.2 Restaurieren einer Stimme

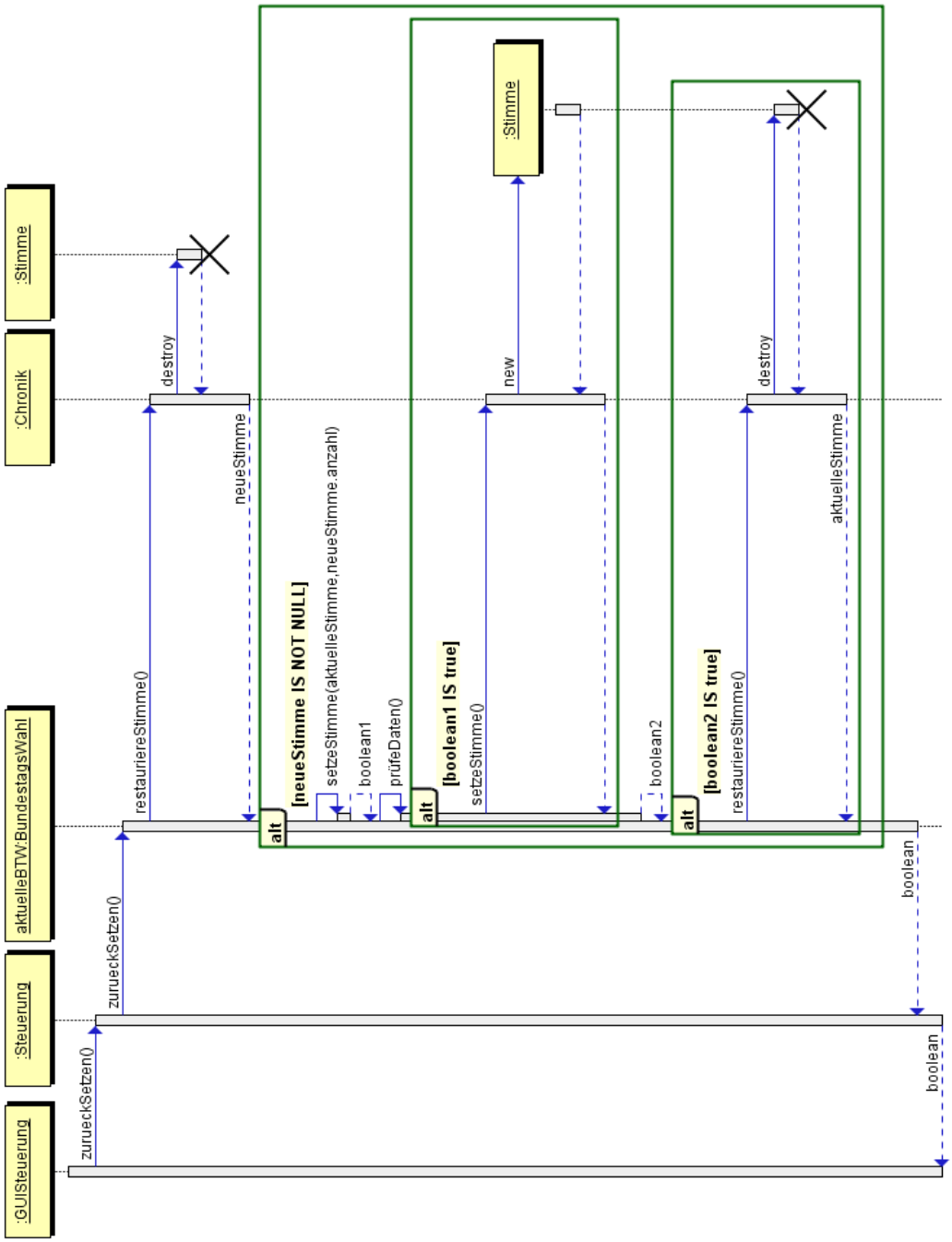

Abbildung 18: Sequenzdiagramm zur Wiederherstellung eines Stimmenobjektes

Stimmen werden in der Chronik Stack-artig zurückgegeben. Sobald eine Bundestagswahl rückgängig gemacht wurde, wird entweder true im Falle einer erfolgreichen Operation, oder false bei einem Fehlschlag zurückgegeben.

4.7 GUI

4.7.1 Aktualisierung

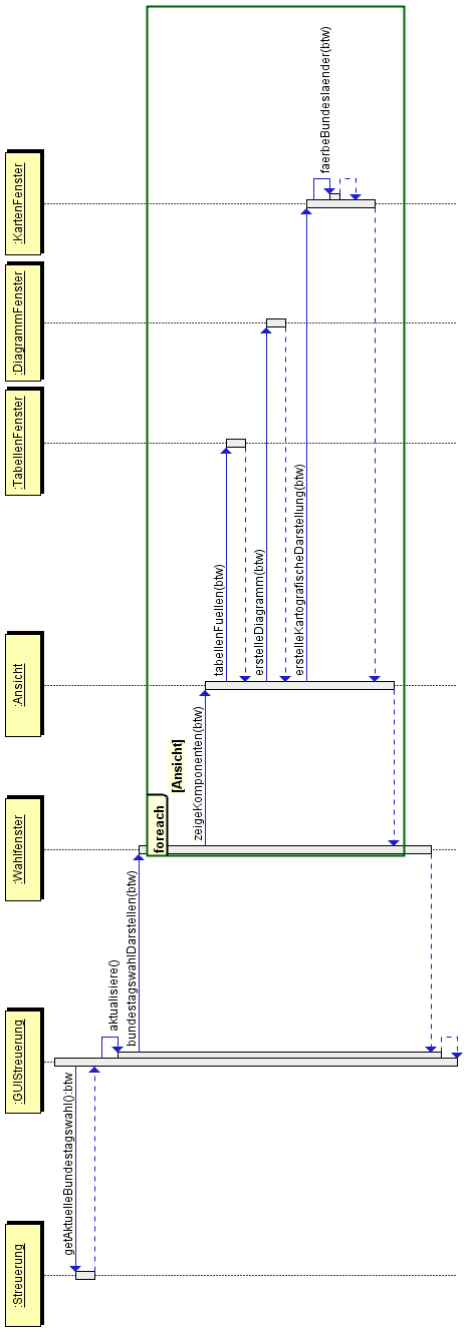

Abbildung 19: Sequenzdiagramm zur Aktualisierung in der GUI

Hier sieht man die Aktualisierung der grafischen Benutzeroberfläche. Eine neue **`Sitzverteilung`** wurde berechnet und in Form einer **`BTW-Klasse`** in der Steuerung abgelegt.
Die **`GUISteuerung`** aktualisiert ihr Bundeswahlobjekt, indem sie sich dieses von der Steuerung holt. Dann wird die Aktualisierung mit der Methode `aktualisiereWahlfenster()`

gestartet. In dieser wird dem **`WahlFenster Bundestagswahl`** übergeben und dieses verteilt die Visualisierung an die drei Ansichten. In diesen wird dann die Datendarstellung and die drei Fenster verteilt. In diesem Fall wird eine kartografische Ansicht erstellt, wodurch auch die private Methode *`faerbeBundeslaender()`* verwendet wird.

4.7.2 Stimmenänderung

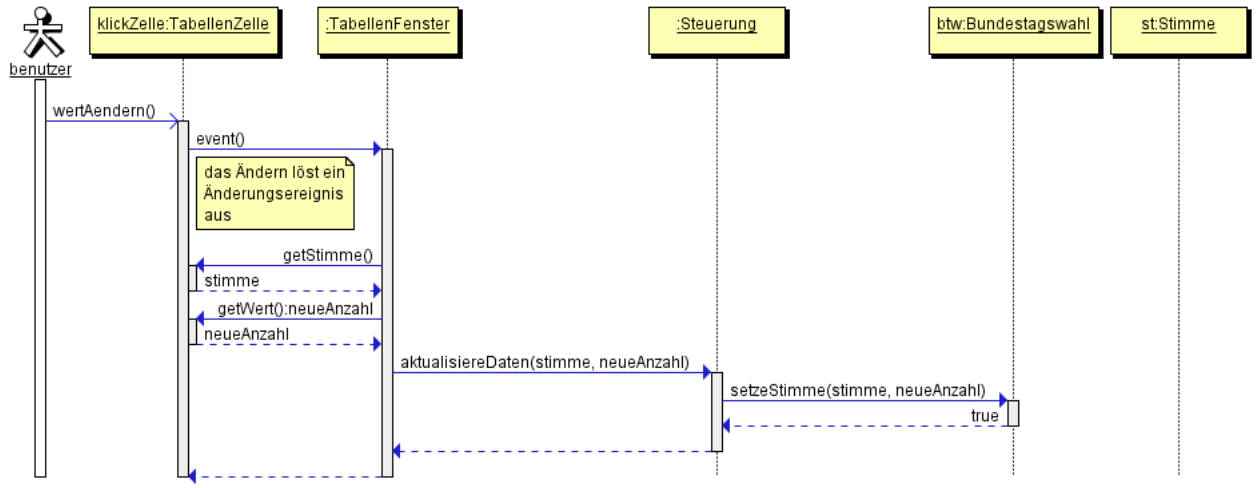

Abbildung 20: Sequenzdiagramm zur Stimmenänderung

Der Benutzer ändert eine beliebige **`Stimme`** im **`TabellenFenster`**. Dadurch wird ein Ereignis ausgelöst, wodurch das **`TabellenFenster`** reagiert. Es fragt die zur **`TabellenZelle`** korrespondierende **`Stimme`** und den neuen Wert ab und übergibt diese zwei Daten an die **`Steuerung`**. Die **`Steuerung`** ruft auf dem aktuellen **`BTW`**-Objekt die Methode setzeStimme() mit den Parametern **`Stimme`** und der neuen Anzahl an Stimmen. Was dabei passiert siehe **`Chronik`**. Alles läuft gut ab, die alte **`Stimme`** wird in die **`Chronik`** aufgenommen und die alte **`Stimme`** in dem **`BTW`**-Objekt durch die alte ersetzt.

5 Implementierungsphasen-Zeitplan

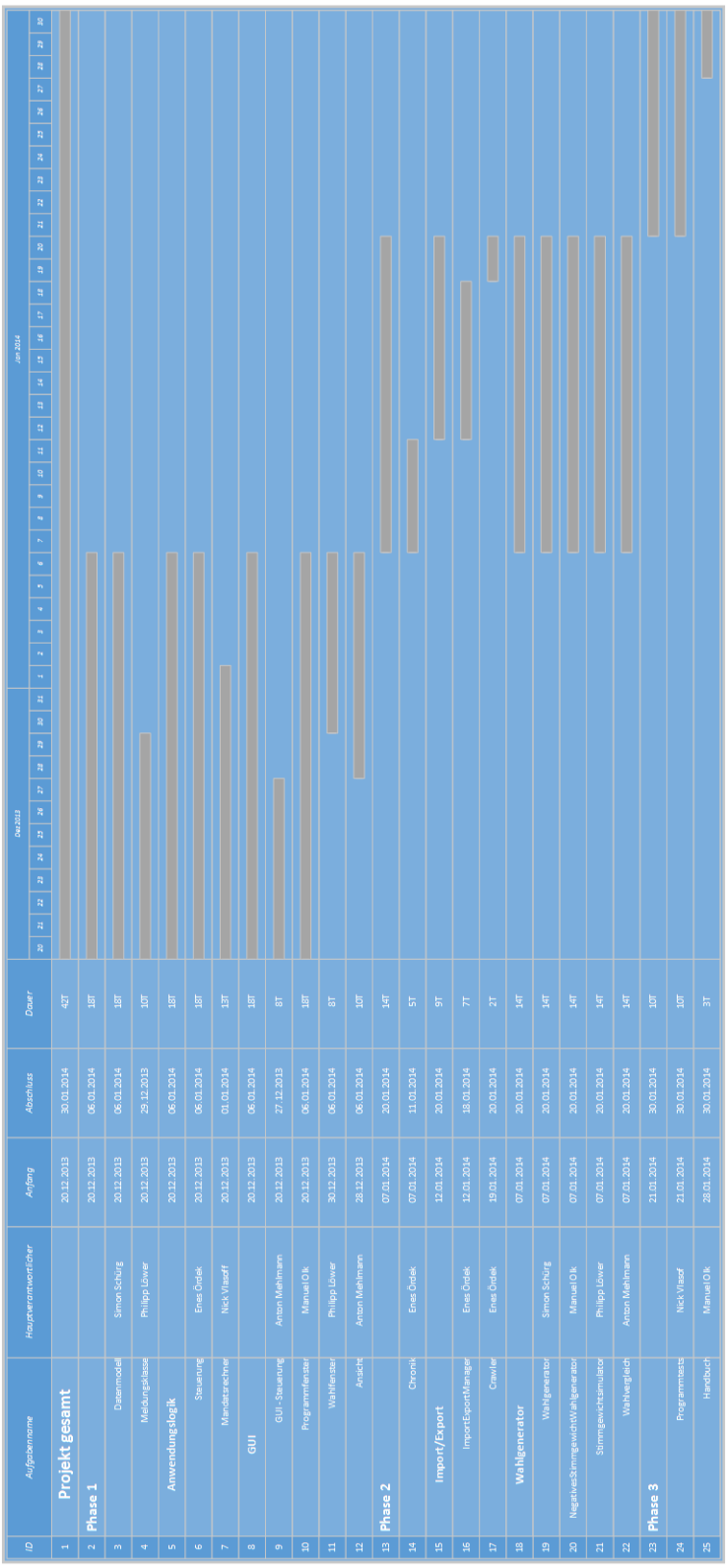

Abbildung 21: Gantt-Diagramm zur Zeitplanung der Implementierungsphase

Die Zeitplanung der Implementierungsphase, sowie bei der Implementierung bedeutende Meilensteine werden in obigem Gantt-Diagramm dargestellt. Zusätzlich ist zu jeder Aufgabe innerhalb der Phase ein Hauptverantwortlicher angegeben.