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

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.
Desweiteren 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.

2 Systemmodell

3 Klassendiagramm

3.1 GUI

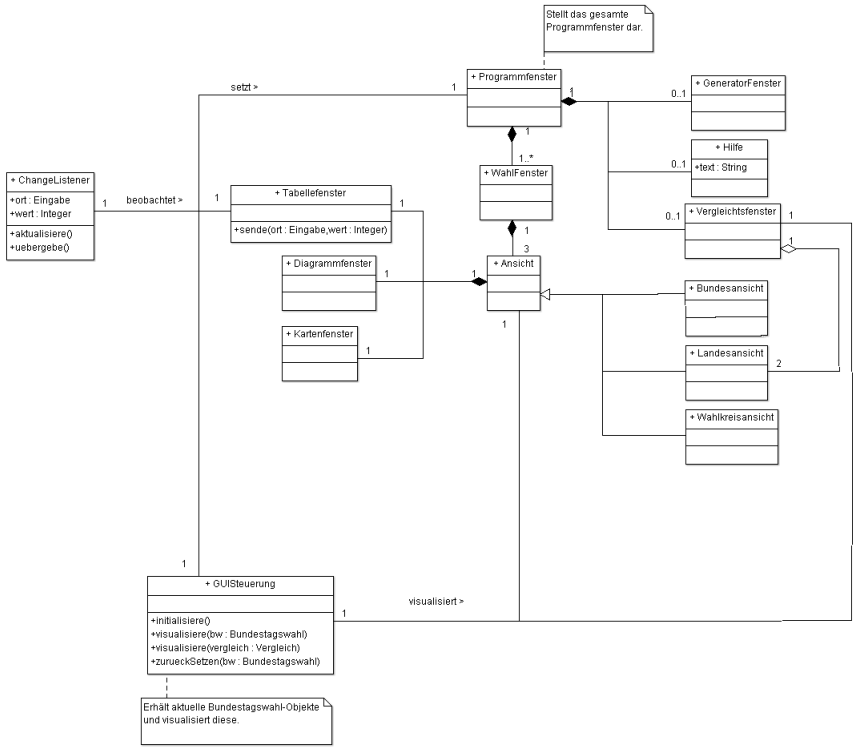


Abbildung 1: GUI Komponente

Dieser Teil des gesamten Klassendiagramm zeigt den Visualisierungsteil.

Das **`Programmfenster`** besteht aus einem oder mehreren **`Wahlfenstern`** in Form von Tabs. Weiterhin enthält es ein **`Generatorfenster`**, um nicht identifizierten Parteien Stimmen zu geben, die Hilfe, und ein Vergleichsfenster, wenn der Vergleich zweier Wahlen aktiv ist.
Ein Wahlfenster hat eine bestimmte Ansicht, diese ist entweder die Bundes-, Landes- und Wahlkreisansicht. Zu jeder von diesen gehört jeweils ein Tabellen-, Diagramm- und Kartenfenster.
Das Tabellenfenster zeigt die Daten der Bundestagswahl-Klasse an (Erst-, Zweitstimmen, Direktmandat,...).
Das Diagrammfenster visualisiert die Sitzverteilungs-Klasse.
Das Kartenfenster visualisiert die Deutschland-Klasse.
Das Tabellenfenster, in welchem Erst- und Zweitstimmen geändert werden können, und die dazugehörige ChangeListener-Klasse werden im Beobachter-Prinzip umgesetzt. Der ChangeListener hört das Tabellenfenster ab und wird bei Veränderungen informiert.
Gesteuert werden alle genannten Klassen durch die GUISteuerungs-Klasse, die die Ansicht und das Vergleichsfenster visualisiert und das gesamte Programmfenster beim Programmstart initialisiert.

Methoden

Tabellenfenster

- **senden(aenderungszeile : String, wert : Integer)**
  Das Tabellenfenster informiert den ChangeListener darüber, welche Zeile im Tabellenfenster geändert wurde und was der neue Wert ist.

ChangeListener

- **aktualisiere()**
  Der ChangeListener passt seine Attribute, nachdem er von dem Tabellenfenster informiert wurde, an, diese repräsentieren den zuletzt geänderte Tabelleneintrag.

- **uebergeben()**
  Der ChangeListener übergibt, nachdem seine Attribute neu gesetzt wurden, diese an die Steuerung, sich dann um die Aktualisierung der internen Daten kümmert.

GUISteuerung

- **initialisiere()**
  Der Programmfensterkonstruktor wird aufgerufen, alle Objekte erstellt und mit der Wahl 2013 befüllt.

- **visualisiere(bw : Bundestagswahl)**
  Wurde eine neue Bundestagswahl geladen und berechnet, werden alle dazugehörigen Werte und Grafiken im Karten-, Diagramm- und Tabellenfenster geladen.

- **visualisiere(vergleich : Vergleich)**
  Hat die Steuerung einen Vergleich zweier Wahlen errechnet und die Daten an die GUISteuerung übergeben, wird das Vergleichsfenster erstellt und mit den Daten des Vergleichsobjektes gefüllt.

- **zurueckSetzen(bw : Bundestagswahl)**
  Wünscht der Benutzer einen zuvor geänderten Wert im Tabellenfenster zurückzusetzen, wird die Steuerung informiert.

3.2 Datenhaltung

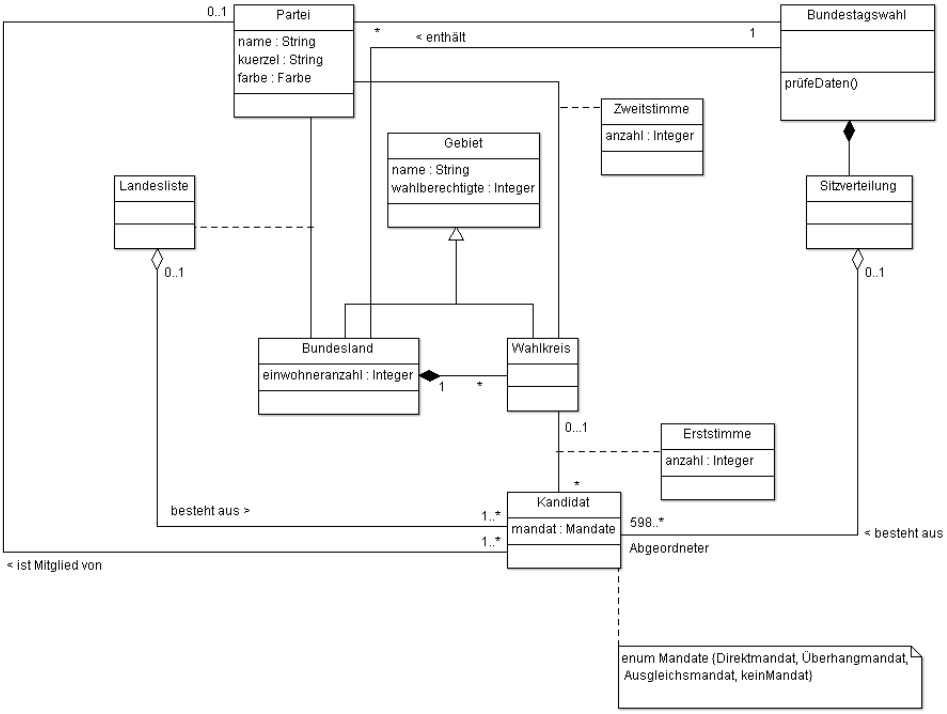


Abbildung 2: Datenhaltungs Komponente

Dieser Ausschnitt des Klassendiagramms stellt den Datenhaltungsteil dar.

3.3 Import/ Export

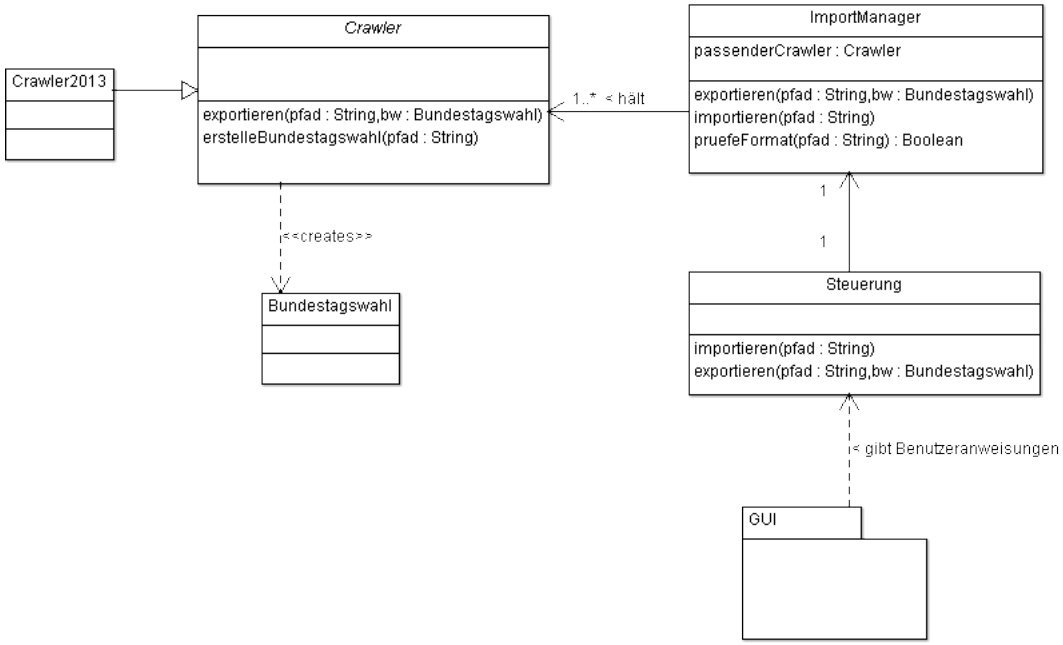


Abbildung 3: Import/Export Komponente

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.

Methoden

`ImportExportManager`

`importieren(csvDatei : Datei) : Bundestagswahl`
Diese öffentliche Methode führt zuerst die private Methode *`pruefeDateityp(csvDatei : Datei) : Boolean`* aus. Wenn diese true zurückgibt, wird die ebenfalls private Methode *`leseCSVDatei(csvDatei : String) : Boolean`* ausgeführt.

Wurde nun eine gültige Bundestagswahl zurückgegeben, wird diese an die Steuerung zurückgegeben, andernfalls ein Fehler ausgegeben.

exportieren(pfad : String)

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 gewonnen Informationen eine Bundestagswahl zu erstellen und zu füllen. Gelingt es einem Crawler, wird der Durchlauf abgebrochen und die gerade erstellte Bundestagswahl wird an den ImportExportManager zurückgegeben.

3.4 Wahlgenerierung

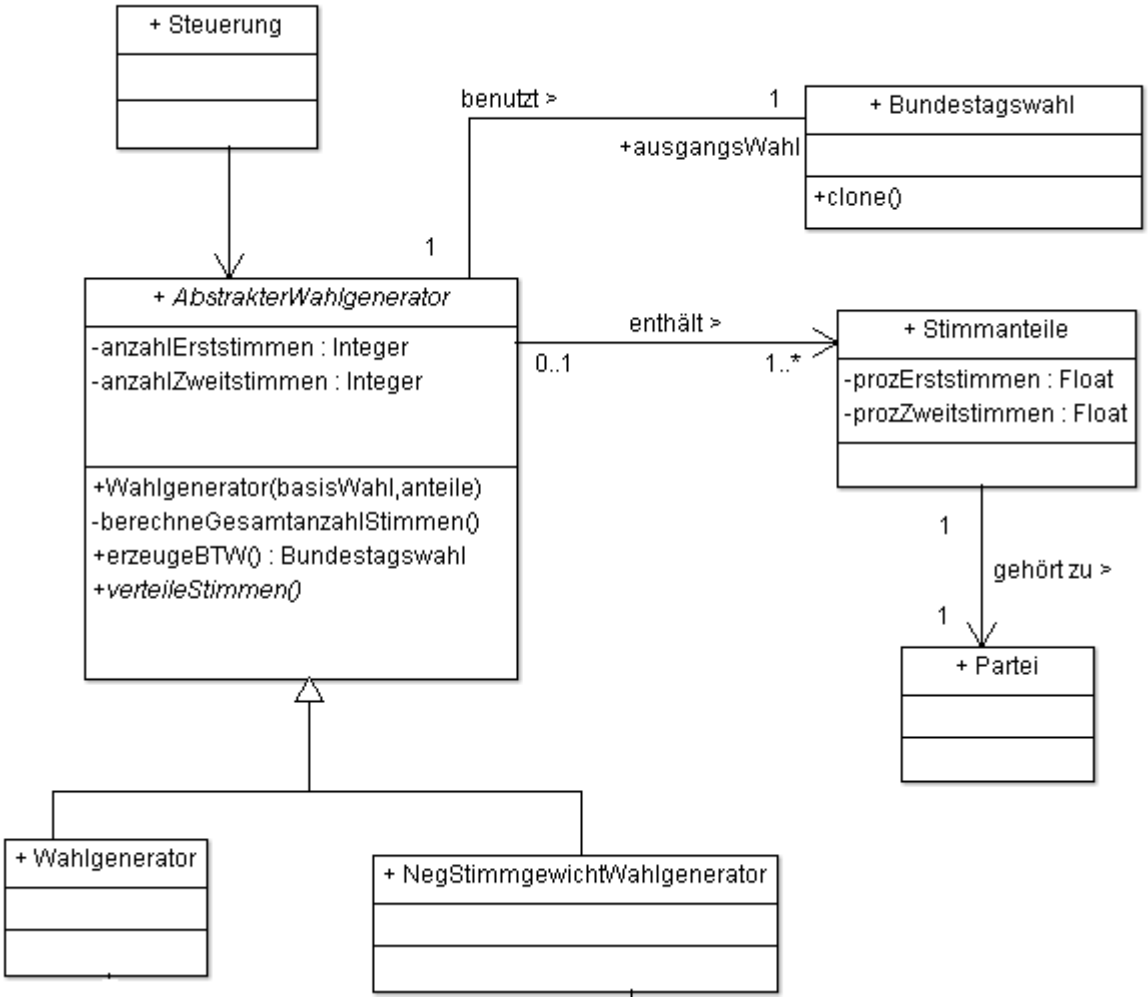


Abbildung 4: Wahlgenerierungs-Komponente

Der obige Ausschnitt des Klassendiagramms zeigt das Wahldatengenerierungs-Modul.

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 definierten **`Stimmanteilen`** auf Bundesebene generiert werden. Bei **`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.

Methoden

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.

3.5 Chronik

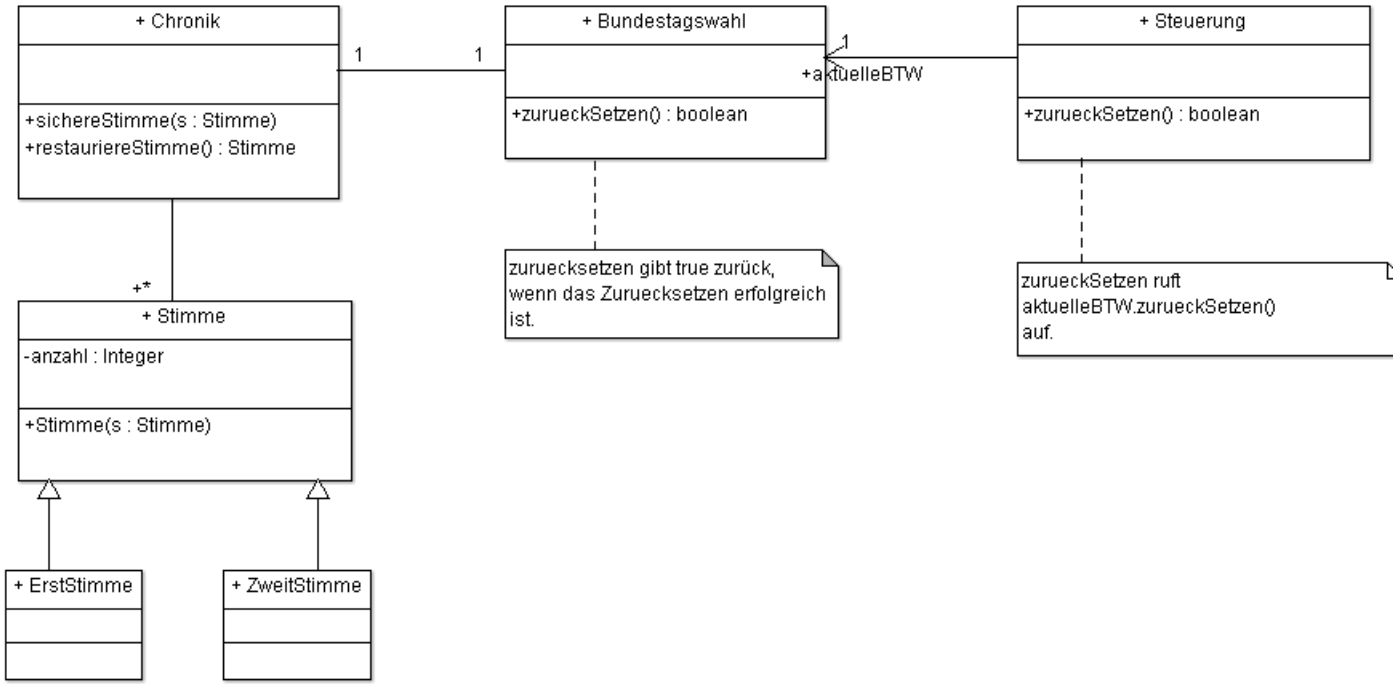


Abbildung 5: Chronik-Komponente

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.

3.6 Mandatsrechner

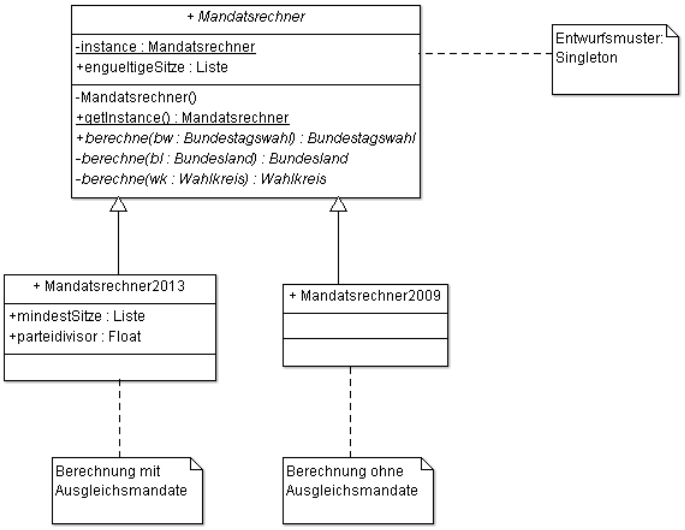


Abbildung 6: Mandatsrechner-Komponente

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. Methoden

`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(bw: 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.

3.7 Meldung

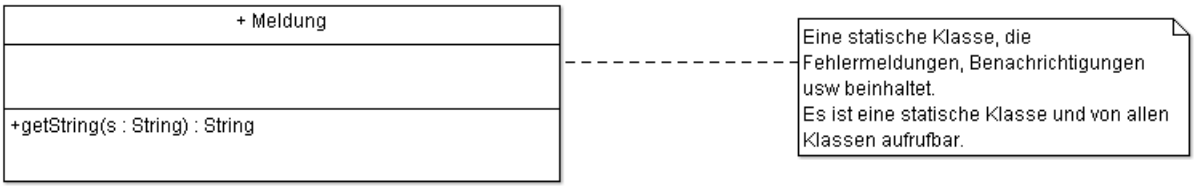


Abbildung 7: Meldungs-Klasse

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 Import

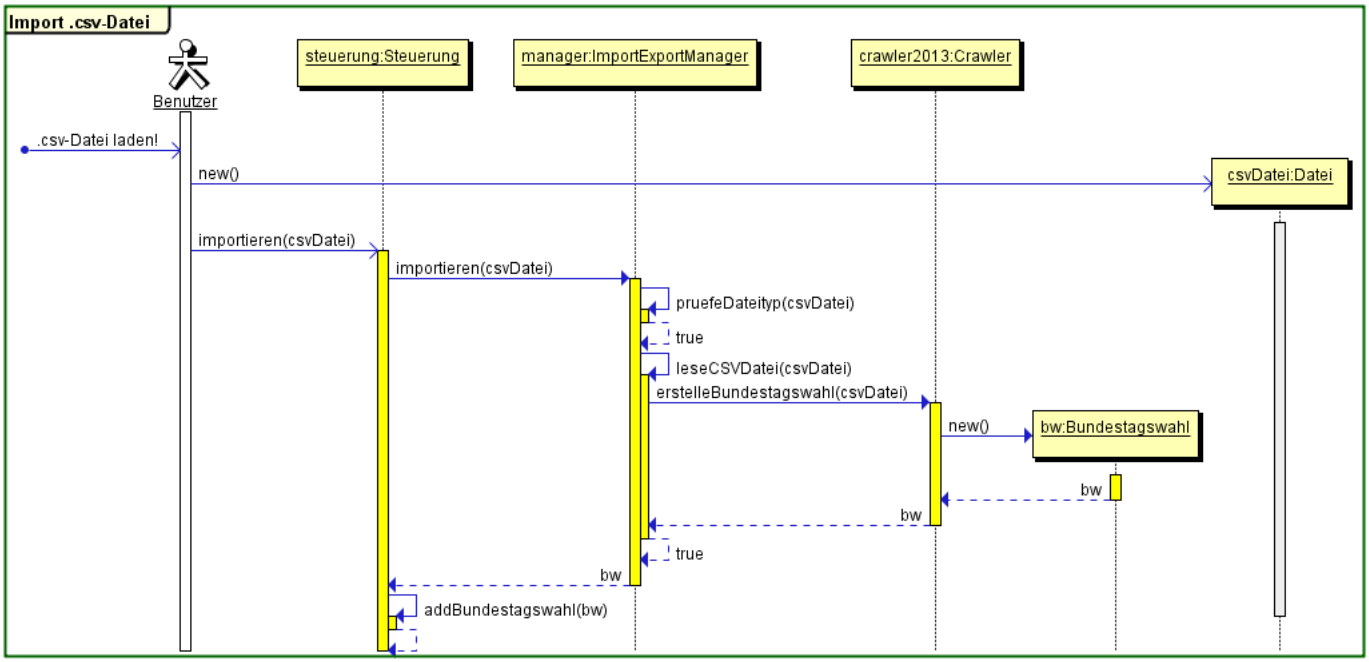


Abbildung 8: Import Sequenzdiagramm

4.2 Wahlgenerierung

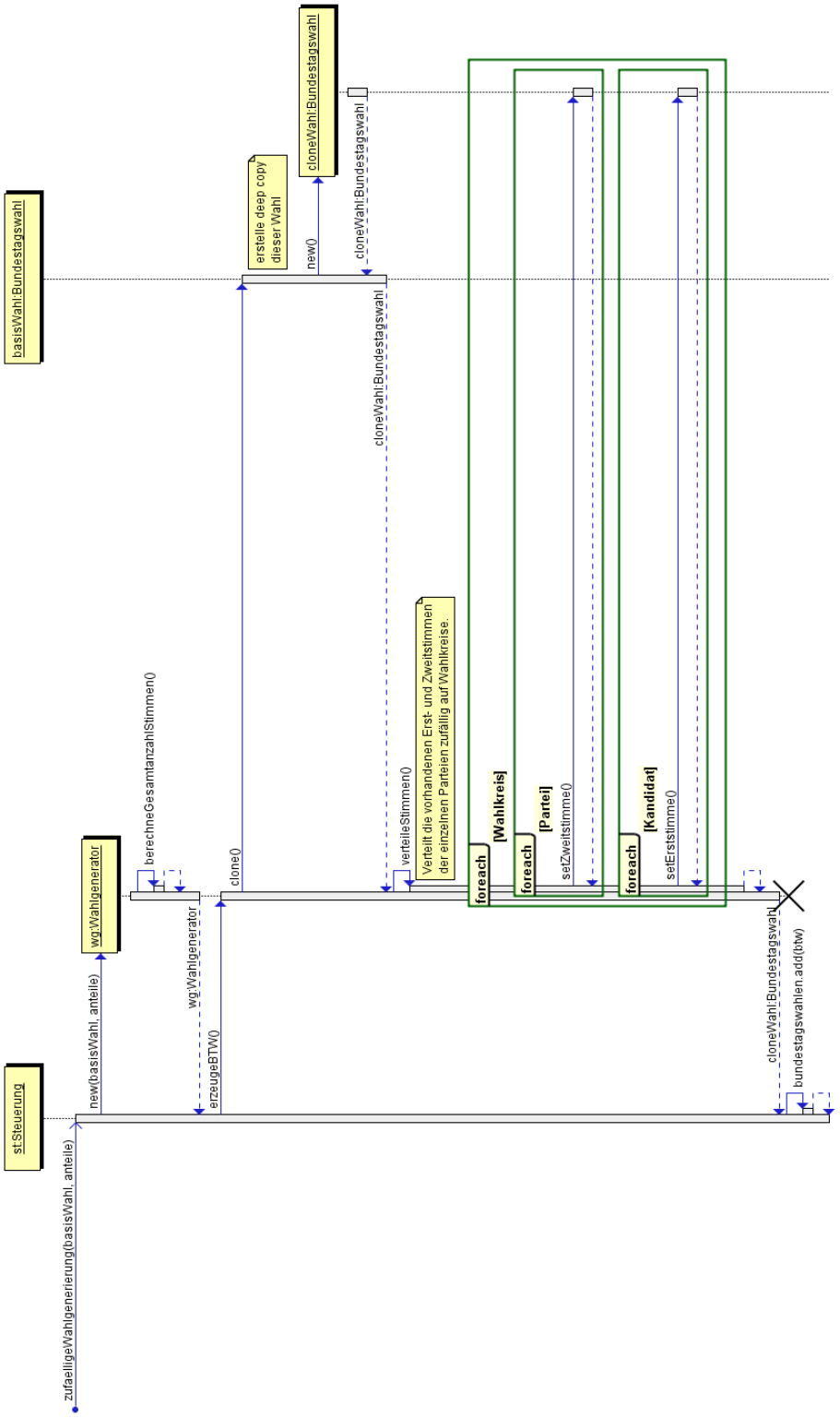


Abbildung 9: Sequenzdiagramm zur Wahlgenerierung

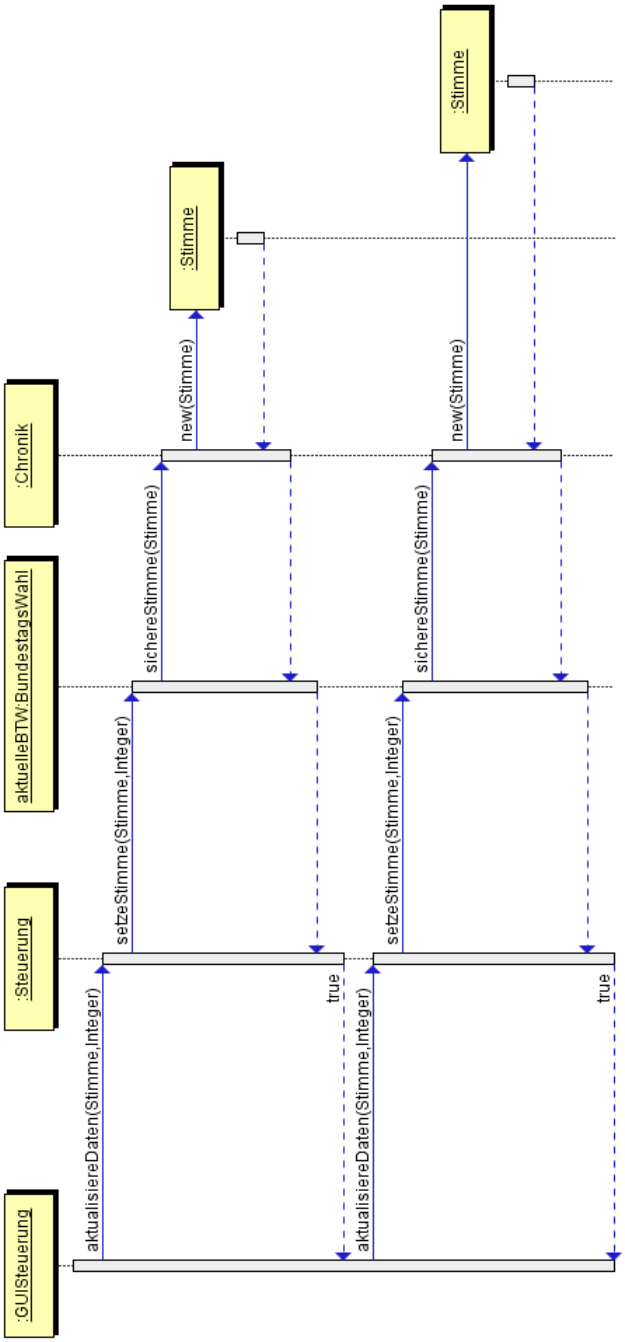
In der Methode *zufaelligeWahlgenerierung(basisWahl : Bundestagswahl, anteile : Stimmanteile) : Bundestagswahl* wird zuerst ein neuer Wahlgenerator erzeugt. In dem Konstruktor des Wahlgenerators werden die absoluten Stimmzahlen berechnet und alle Attribute gesetzt.
Anschließend wird von der Steuerung 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.

4.3 Paradoxe Wahlgenerierung und Vergleich

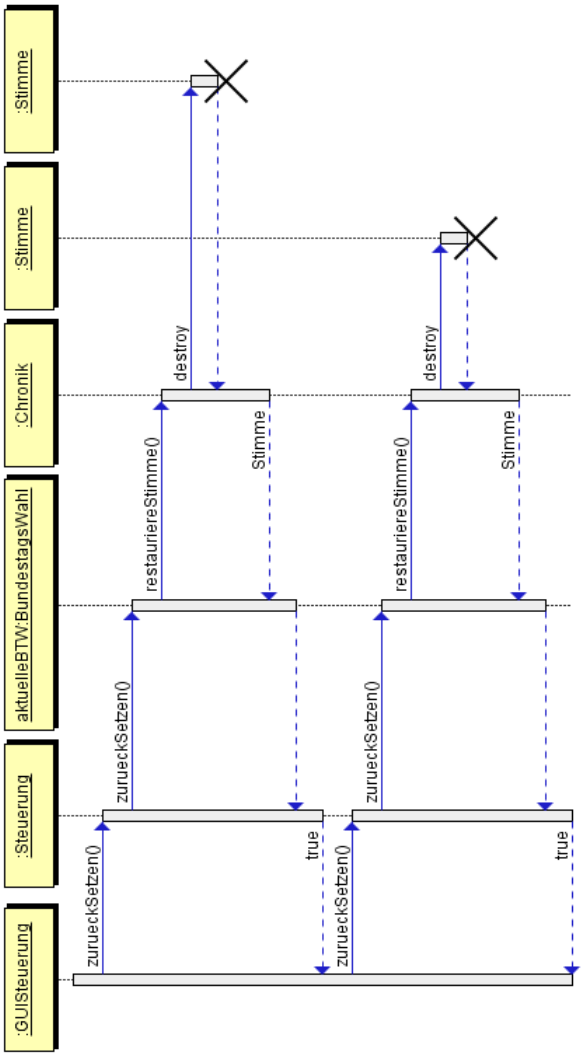
4.4 Vergleich

4.5 Chronik

- Veränderung an den Stimmen



- Restaurieren einer Stimme



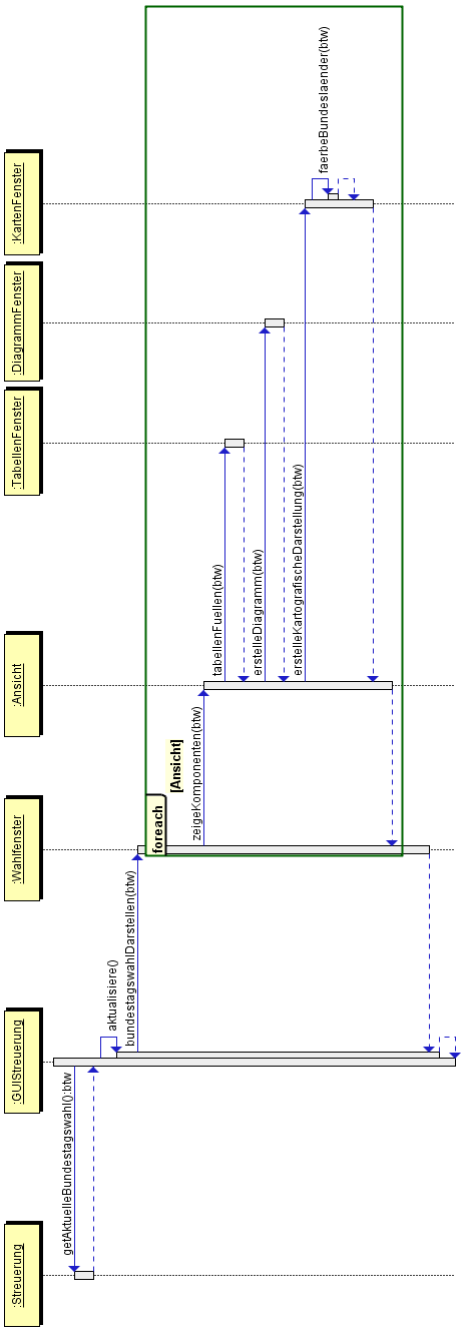
  Stimmen werden in der Chronik werden Stack-artig zurückgegeben. Sobald eine Bundestagswahl rückgängig gemacht wurde, wird die neue Bundestagswahl als Rückgabewert zurückgegeben.

- Beispielszenario

4.6 GUI

4.6.1 Aktualisierung

Hier sieht man die Aktualisierung der grafischen Benutzeroberfläche. Eine neue Sitzverteilung wurde berechnet und in Form einer Bundestagswahl-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 Ansicht. 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.

5 Implementierungsphasen-Zeitplan

Nachfolgendes Gantt-Diagramm stellt die Zeitplanung für den Verlauf der Implementierung des Produktes sowie zugehörige Meilensteine und Hauptverantwortliche dar.