SWE
Systementwurf

Final

ausgearbeitet von:
Erb/Seidel
Tammen/Eck

Fachbereich	:	Informatik
Studiengang	:	SE / TI
Semester	:	5
Datum	:	01.12.2005

1 Festlegung der Einflussfaktoren auf die Softwarearchitektur

1.1 Einsatzbedingungen

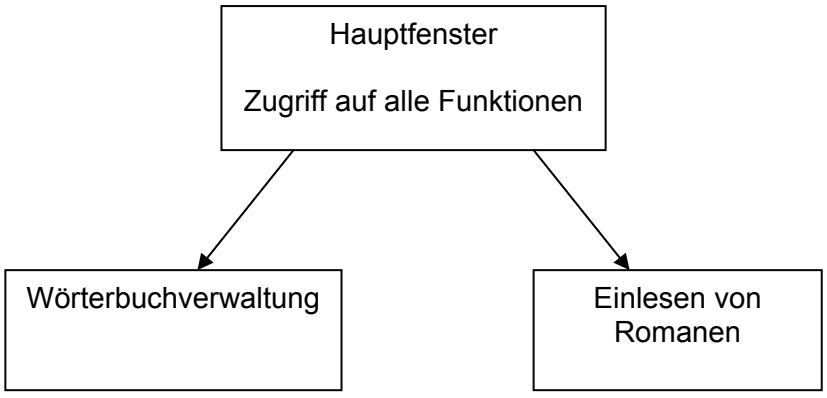
Das Programm wird als Java-Applikation für Einzelplatzrechner ausgeliefert und setzt ein Betriebssystem mit einer installierten Java Runtime Environment für die Benutzung voraus (Java ist zurzeit für Microsoft Windows, OSX und Linux unter http://java.sun.com verfügbar). Des Weiteren wird eine Java Version mit der Installation ausgeliefert. Als mindest Hardwarevoraussetzung wird ein Office-Computer mit mindestens 800MHz und 10mb freien Festplattenspeicherplatz benötigt.

1.2 Nichtfunktionale und Qualitätsanforderungen

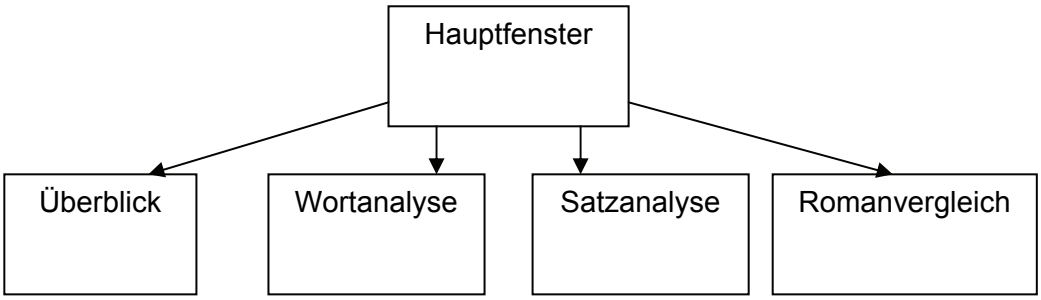
Die Software bietet keine Mehrsprachigkeit in Bezug auf die Benutzeroberfläche. Jedoch können Wörterbücher durchaus in anderen Sprachen verfasst sein, womit auch fremdsprachige Romane analysiert werden können.
Skalierbarkeit wird nicht geboten, da Analysen stets nach dem gleichen Muster erfolgen.
Eine Änderung ist in nächster Zukunft nicht geplant.

2 Entwurf der Benutzerschnittstelle

2.1 Übersicht der Fenster

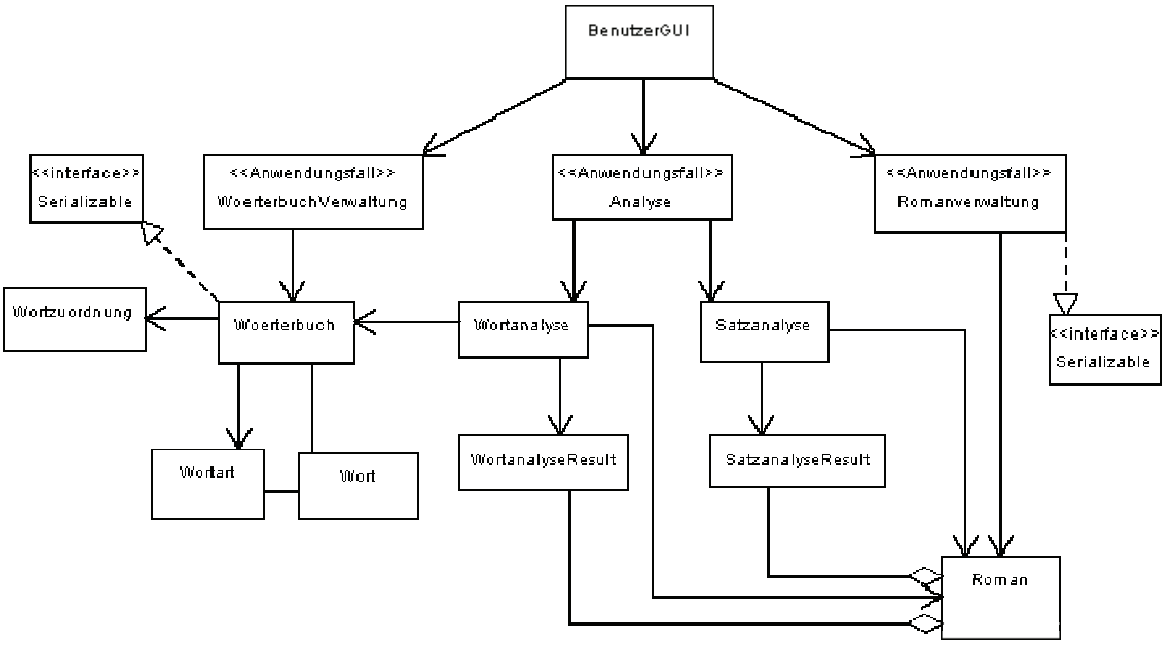


2.2 Übersicht der Tabs des Hauptfensters



3 Grobarchitektur

3.1 Überblick über die Klassen (Schichten)



3.2 Grobbeschreibung der Klassen

3.2.1 Klasse BenutzerGUI

Definition:

Die Klasse BenutzerGUI dient als grafische Benutzerschnittstelle. Über diese sind alle Funktionen ausführbar.

3.2.2 Klasse WoerterbuchVerwaltung

Definition:

In der WoerterbuchVerwaltung können Wörter zu einer Wortart hinzugefügt oder gelöscht werden. Es kann nach einem Wort in allen Wortarten gesucht und eine komplette Datei zu einer Wortart hinzugefügt werden.

3.2.3 Klasse Analyse

Definition:

Die Analyse ruft die Wort- und Satzanalyse auf.

3.2.4 Klasse Romanverwaltung

Definition:

Die Romanverwaltung dient zur Verwaltung der Romane, vom Anlegen bis zum Löschen eines Romans.

3.2.5 Klasse Wortanalyse

Definition:

Die Wortanalyse ermittelt die geforderten Wortanalysen, d.h. durchschnittliche Wortlänge usw.

3.2.6 Klasse Satzanalyse

Definition:

Die Satzanalyse ermittelt die Satzspezifischen Analyseergebnisse, z.B. die kleinste Satzlänge usw.

3.2.7 Klasse Roman

Definition:

Die Klasse Roman hält den Romantext sowie die zu ihm gehörenden Analyseergebnisklassen.

3.2.8 Klasse Wörterbuch

Definition:

Diese Klasse implementiert das Wörterbuch und stellt die entsprechenden Methoden und Datenstrukturen für das Verwalten der Wörter zur Verfügung. Die Klasse wird benutzt von den Klassen WoerterbuchVerwaltung, WortAnalyse und Wort. Sie selbst benutzt die Klasse Wortart sowie WortZuordnung.

3.2.9 Klasse Wortanalyseresult

Definition:

Hier sind alle Ergebnisse der Wortanalyse gespeichert.

3.2.10 Klasse Satzanalyseresult

Definition:

Hier sind alle Ergebnisse der Satzanalyse gespeichert.

3.2.11 Klasse Wortart

Definition:

Mithilfe dieser Klasse wird die im Pflichtenheft festgelegte Klassifizierbarkeit von Wörtern des Wörterbuchs ermöglicht. Ein Wort kann beliebig vielen Wortarten (z.B. „Substantiv“, „Science-Fiction-Helden“) zugewiesen werden und einer Wortart können ebenso beliebig viele Wörter zugewiesen werden.

3.2.12 Klasse Wort

Definition:

Ein Wort besteht aus einer Menge von Zeichen, welche in der Klasse in einem String-Datencontainer gespeichert werden. Beim Einlesen eines Wortes aus dem Quelltext muss zunächst überprüft werden, ob das Wort bereits im Wörterbuch vorhanden ist und wenn dies der Fall ist, welchen Wortarten es zugeordnet ist. Dabei ist es vorteilhaft, stets die Verknüpfungen zwischen Wort und Wortarten präsent zu haben. Daher enthält ein Wort in einem entsprechenden Referenzen auf die zugehörigen Wortarten.

3.2.13 WortZuordnung

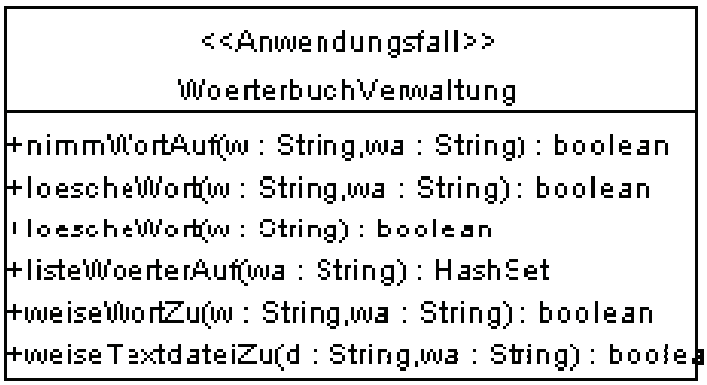
Definition:

Diese Klasse repräsentiert die Zuordnung eines Wortes zu seinen Wortarten. Sie ist notwendig, um die Zuordnung der jeweiligen Wörter im Text getrennt von den Zuordnungen des Wörterbuches halten zu können. Dies ist sinnvoll, da die Wortanalyse nur die Zuordnungen der im Text vorkommenden Wörter benötigt.

4 Feinarchitektur

4.1 Beschreibung der Klassen und ihrer Schnittstellen

4.1.1 Klasse WoerterbuchVerwaltung



4.1.1.1 nimmWortAuf(String wort, String wortart)

Eingabe Parameter:

String wort:	Das eigentliche Wort
String wortart:	Die zum Wort gehörende Wortart

Beschreibung:

Wörter die noch nicht spezifiziert sind, werden in der Wortart „Sonstige Wörter“ gespeichert.

4.1.1.2 loescheWort(String wort, String art)

Eingabe Parameter:

String wort:	Das eigentliche Wort
String wortart:	Die zum Wort gehörende Wortart

Beschreibung:

Löscht das Wort in der übergebenen Wortart.

4.1.1.3 loescheWort(String wort)

Eingabe Parameter:

String wort:	Das eigentliche Wort

Beschreibung:

Löscht das Wort in allen Wortarten.

4.1.1.4 listeWoerterAuf(Wortart art)

Eingabe Parameter:

Wortart art:	Die Wortart deren Wörter aufgelistet werden sollen.

Beschreibung:

Listet die Wörter der Wortart auf.

4.1.1.5 weiseWortZu(Wort wort, Wortart art)

Eingabe Parameter:

Wort wort:	Das eigentliche Wort
Wortart art:	Die zum Wort gehörende Wortart

Beschreibung:

Das Wort wird der Wortart zugewiesen.

4.1.1.6 weiseTextdateiZu(String dateiname, Wortart art)

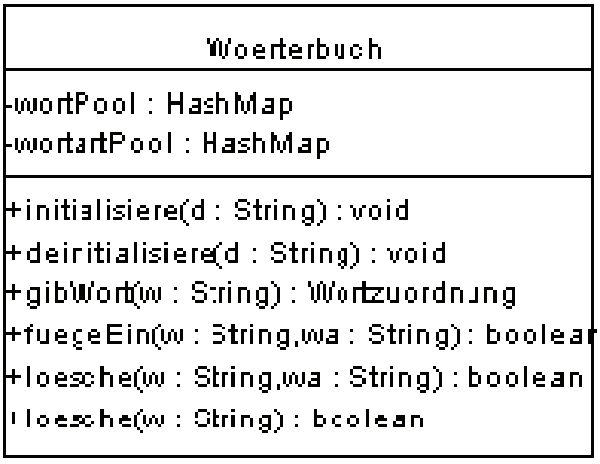
Eingabe Parameter:

String dateiname:	Name der Datei
Wortart art:	Die Wortart, die zur importierenden Datei gehört.

Beschreibung:

Durch Öffnen der gewünschten Textdatei werden die Wörter einer Wortart zugewiesen.

4.1.2 Klasse Woerterbuch



4.1.2.1 initialisiere(String dateiname)

Eingabe Parameter:

String dateiname: Name der Datei

Beschreibung:

Einlesen der Datei in die Datenstruktur

4.1.2.2 deinitialisiere(String dateiname)

Eingabe Parameter:

String dateiname: Name der Datei

Beschreibung:

Speichern der Datenstruktur in die Datei

4.1.2.3 gibWort(String wort)

Eingabe Parameter:

String wort: Gesuchtes Wort

Rückgabe Parameter:

Wortzuordnung: Gefundenes Wort mit dazugehörigen Wortarten

Beschreibung:

Gibt bei Erfolg die Wortzuordnung zurück. Ist das Wort nicht vorhanden, wird eine Exception ausgelöst.

4.1.2.4 fuegeEin(String wort, String art)

Eingabe Parameter:

String wort: Das einzufügende Wort
String art: Die zum Wort gehörende Wortart

Beschreibung:

Fügt das entsprechende Wort der Wortart hinzu und fügt es in die Datenstruktur ein.

4.1.2.5 loesche(String wort, String art)

Eingabe Parameter:

String wort:	Das zu löschende Wort
String art:	Die zum Wort gehörende Wortart

Beschreibung:

Löscht das Wort aus der Wortart.

4.1.2.6 loesche(String wort)

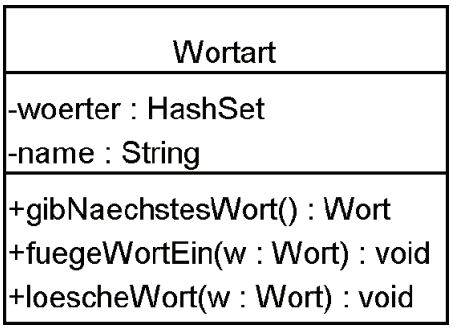
Eingabe Parameter:

String wort:	Das zu löschende Wort

Beschreibung:

Löscht das Wort in jeder Wortart.

4.1.3 Klasse Wortart



4.1.3.1 gibNaechstesWort()

Beschreibung:

Gibt das nächste zu dieser Wortart gehörende Wort zurück

4.1.3.2 fuegeWortEin(Wort wort)

Eingabe Parameter:

Wort wort: Das einzufügende Wort

Beschreibung:

Nimmt das Wort in die Wortart auf.

4.1.3.3 loescheWort(Wort wort)

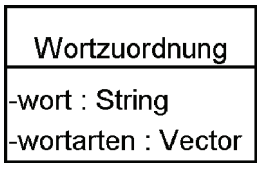
Eingabe Parameter:

Wort wort: Das zu löschende Wort

Beschreibung:

Entfernt das Wort aus der Wortart.

4.1.4 Klasse WortZuordnung



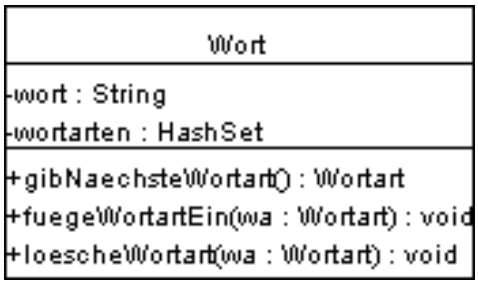
Attribute:

Vector Wortarten: Eine Liste der zum Wort gehörenden Wortarten

Beschreibung:

Diese Klasse repräsentiert die Zuordnung eines Wortes zu seinen Wortarten. Sie ist notwendig, um die Zuordnung der jeweiligen Wörter im Text getrennt von den Zuordnungen des Wörterbuches halten zu können. Dies ist sinnvoll, da die Wortanalyse nur die Zuordnungen der im Text vorkommenden Wörter benötigt.

4.1.5 Klasse Wort



Attribute:

String wort:	Das Wort
HashSet wortarten:	Enthält Verweise auf alle zugehörigen Wortarten

4.1.5.1 gibNaechsteWortart()

Beschreibung:

Gibt die nächste Wortart zu dem Wort zurück

4.1.5.2 fuegeWortartEin(Wortart art)

Eingabe Parameter:

Wortart art:	Hinzuzufügende Wortart

Beschreibung:

Fügt dem Wort eine Wortart hinzu

4.1.5.3 loescheWortart(Wortart art)

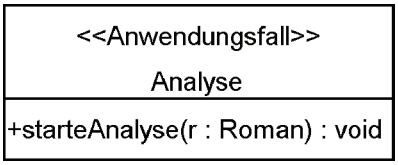
Eingabe Parameter:

Wortart art:	Zu löschende Wortart

Beschreibung:

Löscht die Zuordnung des Wortes zur übergebenen Wortart

4.1.6 Klasse Analyse



4.1.6.1 starteAnalyse()

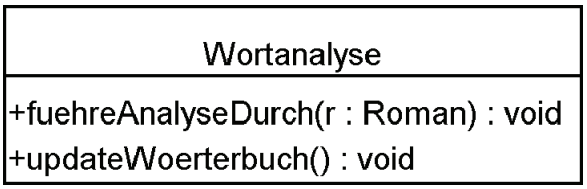
Eingabe Parameter:

Roman roman: Zu analysierender Roman

Beschreibung:

Diese Funktion ruft die Funktionen ***fuehreAnalyseDurch()*** der Klassen Wortanalyse und Satzanalyse auf.

4.1.7 Klasse Wortanalyse



4.1.7.1 fuehreAnalyseDurch(Roman roman)

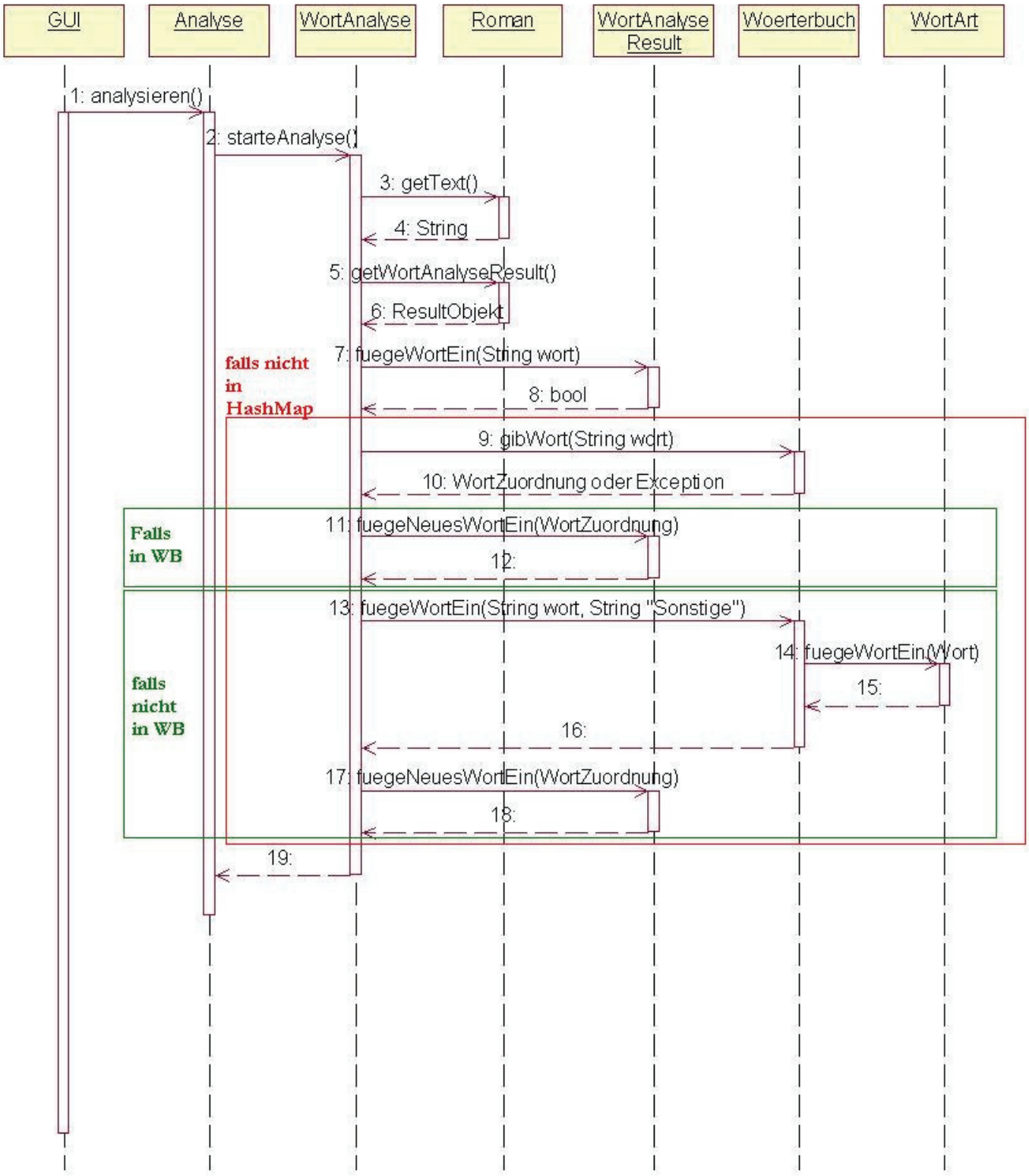
Eingabe Parameter:

Roman roman: Zu analysierender Roman

Beschreibung:

In dieser Funktion findet die Wortanalyse statt. Der übergebene Roman wird Wort für Wort abgearbeitet. Bei jedem Wort wird die Funktion ***fuegeWortEin()*** der Klasse WortanalyseResult aufgerufen. Ist das Wort vorhanden gibt diese Funktion ***true*** zurück. Ist das Wort nicht vorhanden muss im Wörterbuch nach dem Wort gesucht werden. Wird das Wort dort gefunden, muss die zurückgegebne Wortzuordnung in der HashMap WortZuordnung abgespeichert werden. Wird das Wort nicht gefunden muss ein neues Wort sowohl in der Wortart ***„Sonstige Wörter“***, als auch in der HashMap WortZuordnung eingefügt werden.

Sequenzdiagramm WortAnalyse

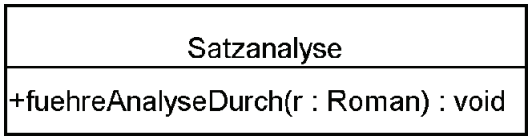


4.1.7.2 updateWoerterbuch()

Beschreibung:

Überträgt, nachdem der Benutzer die „Sonstigen Wörter“ zugeordnet hat, die neuen Zuordnungen ins Wörterbuch.

4.1.8 Klasse Satzanalyse



4.1.8.1 fuehreAnalyseDurch(Roman roman)

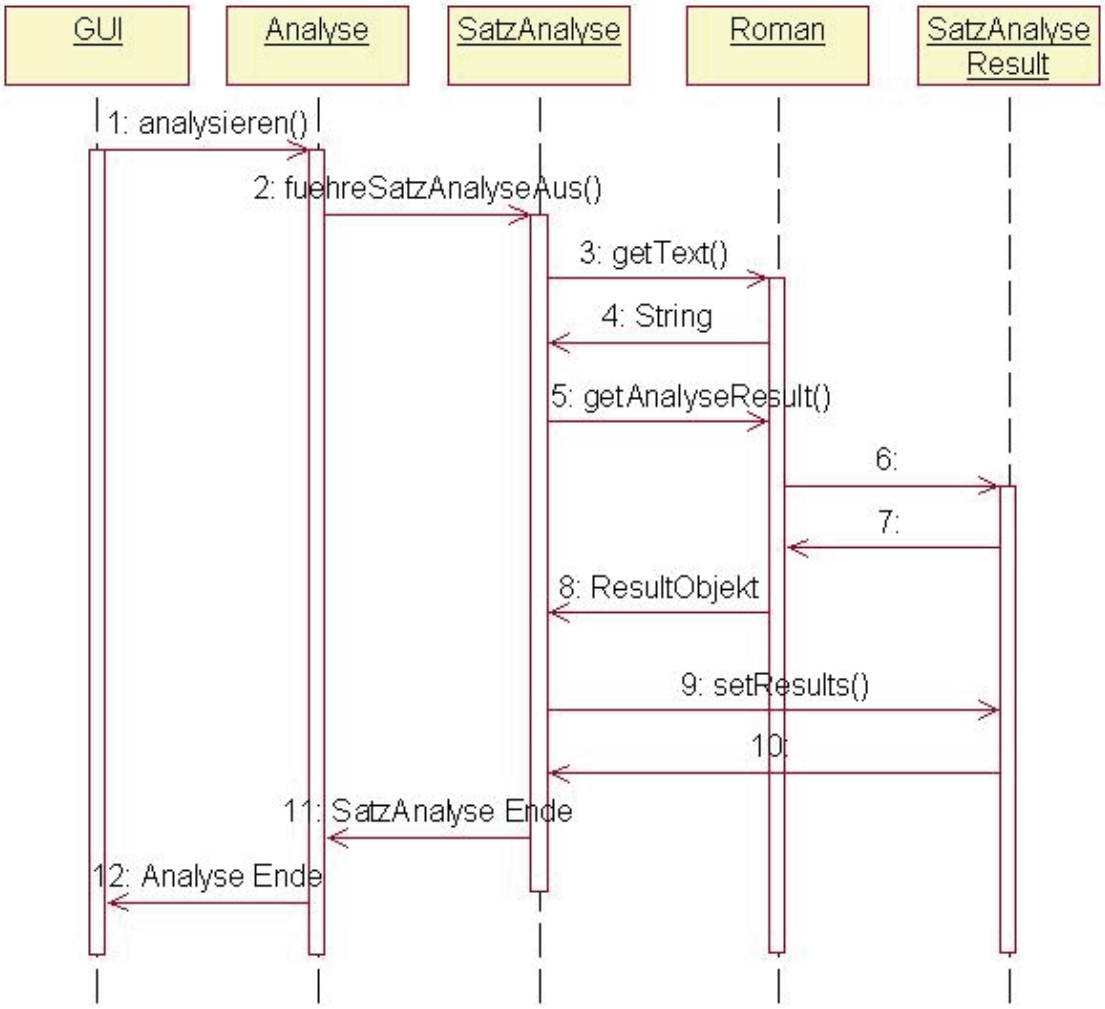
Eingabe Parameter:

Roman roman: zu analysierender Roman

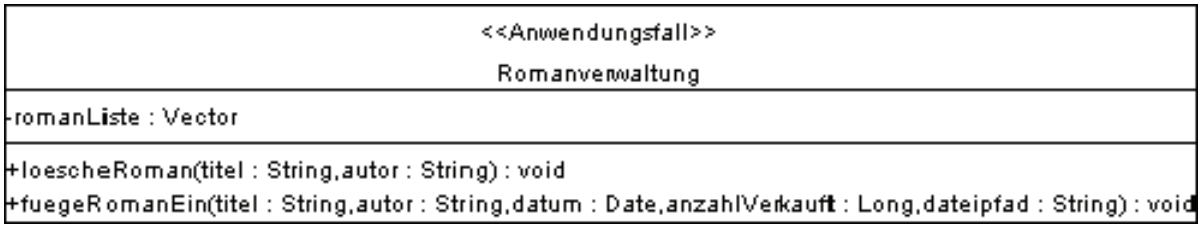
Beschreibung:

Diese Funktion ist zuständig für die Analyse der Sätze. Dazu wird der komplette Romantext durchgegangen und die Länge der Sätze ermittelt, anhand dieser Länge wird dann die durchschnittliche Satzlänge ermittelt. Außerdem wird die Anzahl der Sätze, der Nebensätze und der komplexen Sätze ermittelt und in die dafür vorgesehenen Attribute in der Klasse SatzanalyseResult eingetragen.

Sequenzdiagramm SatzAnalyse



4.1.9 Klasse Romanverwaltung



4.1.9.1 loescheRoman(String autor, String titel)

Eingabe Parameter:

String autor: Autor des Romans
String titel: Name des Romans

Beschreibung:

Diese Funktion löscht den Roman mit dem Namen ***roman*** aus der ***romanListe***. Falls kein Roman mit dem diesem Namen existiert wird ***false*** zurückgegeben.

4.1.9.2 fuegeRomanEin(String titel, String autor, Date datum, long anzahlVerkauft, String dateipfad)

Eingabe Parameter:

String titel: Name des Romans
String autor: Autor des Romans
Date datum: Erscheinungsdatum des Romans
Long anzahlVerkauft: Anzahl der verkauften Romane
String dateipfad: Dateipfad der Roman Textdatei

Beschreibung:

Diese Funktion legt in der ***romanListe*** einen neuen Roman an.

4.1.10 Klasse Roman

Beschreibung:

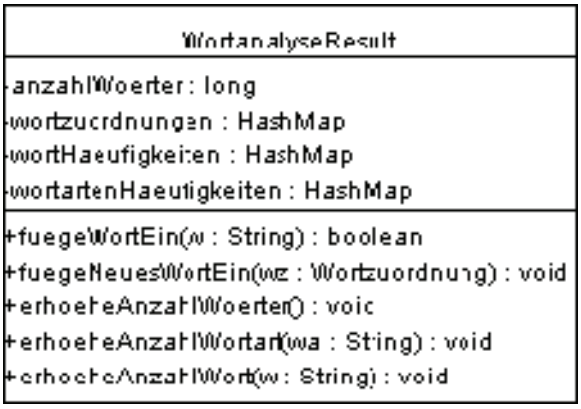
Enthält die Attribute des Romans, sowie die entsprechenden set- und get-Methoden.

4.1.10.1 leseTextEin()

Beschreibung:

Liest den Romantext aus der Datei in den String „text“ ein.

4.1.11 Klasse WortanalyseResult



Attribute:

Long anzahlWoerter: Gesamtanzahl der Wörter
HashMap wortzuordnungen: Datenstruktur die alle Wortarten mit dazugehörigen Wörtern enthält.
HashMap wortHaeufigkeiten: Speichert für jedes Wort dessen Häufigkeit
HashMap wortartHaeufigkeiten: Speichert für jede Wortart deren Häufigkeit

4.1.11.1 fuegeWortEin(String wort)

Eingabe Parameter:

String wort: einzufügendes Wort

Beschreibung:

Diese Funktion sucht nach dem einzufügenden Wort in der HashMap Wortzuordnung. Falls das Wort schon vorhanden ist, zählt es die Worthäufigkeit des Wortes sowie aller zugehörigen Wortarten um eins hoch. Wenn es nicht vorhanden ist, so muss es im Wörterbuch gesucht werden.

4.1.11.2 fuegeNeuesWortEin(Wortzuordnung wz)

Eingabe Parameter:

Wortzuordnung wz: einzufügende Wortzuordnung

Beschreibung:

Fügt die Wortzuordnung ohne Prüfung, ob das Wort bereits vorhanden ist, in die Datenstruktur ein.

4.1.11.3 erhoeheAnzahlWoerter()

Beschreibung:

Erhöht die Anzahl der Wörter (Attribut „anzahlWoerter“) um eins.

4.1.11.4 erhoeheAnzahlWortart(String wa)

Eingabe Parameter:

String wa: Der Name der Wortart

Beschreibung:

Erhöht die Anzahl der übergebenen Wortart in der HashMap „wortartenHaeufigkeiten“ um eins. (Schlüssel: Name der Wortart, Wert: Häufigkeit)

4.1.11.5 erhoeheAnzahlWort(String w)

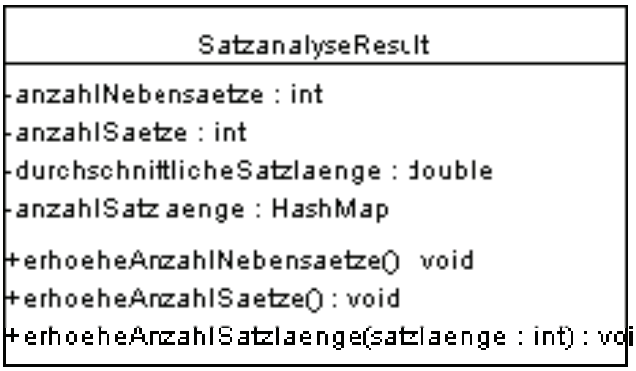
Eingabe Parameter:

String w: Das Wort

Beschreibung:

Erhöht die Anzahl des übergebenen Wortes in der HashMap „wortHaeufigkeiten“ um eins. (Schlüssel: Wort, Wert: Häufigkeit)

4.1.12 Klasse SatzanalyseResult



Attribute:

int anzahlNebensaetze:	Gesamtanzahl der Nebensätze
int anzahlSaetze:	Gesamtanzahl der Sätze
double durchschnittlicheSatzlaenge:	Durchschnittliche Satzlänge
HashMap anzahlSatzlaenge:	Speichert für jede Satzlänge deren Häufigkeit (Schlüssel: Satzlänge, Wert: Häufigkeit)

4.1.12.1 erhoeheAnzahlNebensaetze()

Beschreibung:
Erhöht die Anzahl der Nebensätze (Attribut „anzahlNebensaetze“) um eins.

4.1.12.2 erhoeheAnzahlSaetze()

Beschreibung:
Erhöht die Anzahl der Sätze (Attribut „anzahlSaetze“) um eins.

4.1.12.3 erhoeheAnzahlSatzlaenge(int laenge)

Eingabe Parameter:

int laenge:	Die Satzlänge

Beschreibung:
Erhöht die Anzahl der Sätze mit der übergebenen Länge in der HashMap „anzahlSatzlaenge“ um eins. (Schlüssel: Satzlänge, Wert: Häufigkeit)

Überblick über Klassen und Beziehungen (Module)