

words Package

words:: Syllable
Fields
- onset : String
Properties
+ coda : String + nucleus : String
Constructors
+ Syllable(char) : void + Syllable(String) : void + Syllable(String, String, String) : void + Syllable(Map<String, String>) : void
Methods
+ equals(Object) : boolean + hashCode() : int + rhymes(Syllable) : boolean + setOnset(String) : void + toJsonObject() : JSONObject + toString() : String

«final» «enum» words::SubWord:: PartOfSpeech
Constants
«synthetic» - \$VALUES : PartOfSpeech[] + ADJECTIVE : PartOfSpeech + ADVERB : PartOfSpeech + CONJUNCTION : PartOfSpeech + DEFINITE_ARTICLE : PartOfSpeech + NOUN : PartOfSpeech + PREPOSITION : PartOfSpeech + PRONOUN : PartOfSpeech + UNKNOWN : PartOfSpeech + VERB : PartOfSpeech
Properties
«final» «readOnly» + apiString : String
Constructors
- PartOfSpeech(String, int, String) : void
Methods
+ fromString(String) : PartOfSpeech + valueOf(String) : PartOfSpeech + values() : PartOfSpeech[]

words:: AbstractIPA
Constants
CONSONANTS : Character[] # DIPTHONGS : String[] # ONSETS : String[] # U_ONSETS : String[] # U_VARIANTS : String[] # VOWELS : Character[]
Constructors
+ AbstractIPA() : void
Methods
isConsonant(char) : boolean # isDiphthong(String) : boolean # isValidOnset(String, String) : boolean # isVowel(char) : boolean



words:: IPAHandler
Constants
- P_EMPHASIS : char - SPACE : char - S_EMPHASIS : char
Constructors
+ IPAHandler() : void
Methods
- addMissingEmphasis(String) : String + getSyllables(String) : Pair<ArrayList<Syllable>, Emphasis>

words:: Poem
Fields
- fileReader : BufferedReader
Properties
«readOnly» + lineCount : int «readOnly» + stanzaCount : int «readOnly» + stanzas : ArrayList<Stanza> «readOnly» + string : String + title : String
Constructors
+ Poem(String, String) : void + Poem(Path) : void
Methods
- fillPoem() : void + joinWords(int, int, int, int) : boolean + savePoem(File) : void + setDefaultRhymeScheme(String) : void + splitWord(int, int, int, String) : boolean + substituteWord(int, int, int, SuperWord) : void + toString() : String

words:: Pronunciation
Fields
- all : SubPronunciation - subPronunciations : EnumMap<PartOfSpeech, SubPronunciation>
Constructors
+ Pronunciation() : void
Methods
+ getSubPronunciation(String, PartOfSpeech) : SubPronunciation + getSyllableCount(PartOfSpeech) : int + setIPA(String, String) : boolean + setIPA(String, JSONObject) : boolean + setRhyme(String, String) : boolean + setRhyme(String, JSONObject) : boolean + toString() : String

«memberClass» words::Pronunciation:: SubPronunciation
Fields
~ emphasis : Emphasis ~ ipa : String ~ primaryRhymeSubstring : ArrayList<Syllable> ~ secondaryRhymeSubstrings : ArrayList<ArrayList<Syllable>> ~ syllables : ArrayList<Syllable>
Constructors
+ SubPronunciation() : void
Methods
- forcedRhymesWith(SubPronunciation) : boolean - getRhymeSubstring(int) : ArrayList<Syllable> - imperfectRhymesWith(SubPronunciation) : boolean + matchesWith(Filter, SubPronunciation) : boolean - populateRhymes() : void - rhymeMatch(ArrayList<Syllable>, ArrayList<Syllable>) : boolean - rhymesWith(SubPronunciation) : boolean - syllabicRhymesWith(SubPronunciation) : boolean + toString() : String - weakRhymesWith(SubPronunciation) : boolean

words:: RhymeScheme
Properties
«readOnly» + lineCount : int «readOnly» + nextValue : int «readOnly» + scheme : int[]
Constructors
+ RhymeScheme(char[]) : void + RhymeScheme(int) : void + RhymeScheme(int, char[]) : void + RhymeScheme(int, int[]) : void
Methods
- convertChars(char[]) : int[] - convertInt(int) : char + getValue(int) : int + setValue(int, int) : void + toString() : String

words:: Emphasis
Properties
+ primary : int «readOnly» + secondary : ArrayList<Integer>
Constructors
+ Emphasis() : void
Methods
+ addSecondary(int) : void + equals(Object) : boolean + toString() : String

words:: Token
Properties
«final» «readOnly» + plaintext : String
Constructors
+ Token() : void + Token(String) : void
Methods
+ toString() : String



words:: SuperWord
Fields
- cachePlaceholder : HashMap<String, SuperWord> - cachePopulated : HashMap<String, SuperWord> - knownFields : Set<String> - populated : boolean - subWords : EnumMap<PartOfSpeech, ArrayList<SubWord>>
Properties
«readOnly» + plaintextSyllables : ArrayList<String> «readOnly» + pronunciation : Pronunciation
Constructors
- SuperWord(String) : void
Methods
+ batchPlaceHolders(List<Object>) : ArrayList<SuperWord> - filterSuggestions(ArrayList<SuperWord>, PartOfSpeech, FilterParameters) : ArrayList<SuperWord> - getAggregatedSuggestions(PartOfSpeech, SuggestionPoolParameters) : ArrayList<SuperWord> + getFilteredSuggestions(PartOfSpeech, SuggestionPoolParameters, FilterParameters) : ArrayList<SuperWord> + getSubPronunciation(PartOfSpeech) : SubPronunciation + getSubWords(PartOfSpeech, boolean) : ArrayList<SubWord> + getSuggestionPool(SuggestionPool, PartOfSpeech, boolean) : ArrayList<SuperWord> + getSuperWord(String) : SuperWord + getSyllableCount(PartOfSpeech) : int «synthetic» - lambda\$setSubWords\$0(SubWord, PartOfSpeech, ArrayList) : ArrayList - matchesWith(RhymeType, String, PartOfSpeech, SubPronunciation, String, PartOfSpeech, SubPronunciation) : boolean + matchesWithWrapper(RhymeType, SuperWord) : boolean + populate() : void + rhymesWithWrapper(RhymeType, SuperWord, PartOfSpeech, PartOfSpeech) : boolean - setSubWords(JSONArray) : void - setSyllables(String, JSONObject) : void + subWordsString() : String + toFullString() : String + toString() : String + validPool(SuggestionPool, PartOfSpeech) : boolean

words:: SubWord
Fields
<ul style="list-style-type: none"> - definition : String - knownFields : Set<String> «final» - parent : SuperWord - setCommonCategories : boolean - setCommonlyTyped : boolean - suggestionPools : EnumMap<SuggestionPool, ArrayList<SuperWord>>
Constructors
+ SubWord(SuperWord, Map<String, Object>) : void
Methods
<ul style="list-style-type: none"> + getSuggestionPool(SuggestionPool) : ArrayList<SuperWord> - setCommonCategories() : void - setCommonlyTyped() : void - setPartOfSpeech(String) : void + toString() : String

words:: Stanza
Fields
<ul style="list-style-type: none"> - actualScheme : RhymeScheme - desiredScheme : RhymeScheme
Properties
<ul style="list-style-type: none"> «readOnly» + actualRhymeScheme : RhymingScheme «readOnly» + desiredRhymeScheme : RhymingScheme «readOnly» + lines : ArrayList<ArrayList<Token>> «readOnly» + startLine : int
Constructors
+ Stanza(int) : void
Methods
<ul style="list-style-type: none"> + addLine(String) : void + evaluateRhymingScheme() : void - getLastWord(int) : SuperWord + joinWords(int, int, int) : boolean + lineCount() : int + setDesiredRhymeScheme(String) : boolean + setDesiredRhymeSchemeFromDefault(RhymeScheme) : boolean + splitWord(int, int, String) : boolean + substituteWord(int, int, SuperWord) : void + toString() : String

words Package

words_api:: WordsAPI
Fields
~ cache : HashMap<String, JSONObject> ~ client : HttpClient
Constructors
+ WordsAPI() : void
Methods
- getRequest(URI) : HttpRequest - getUri(String, String) : URI + getWord(String) : JSONObject - sendRequest(URI) : JSONObject

config Package

config:: Configuration
Constants
+ LOG : LogWriter
Constructors
+ Configuration() : void

exceptions Package

exceptions:: RhymeSchemeSizeException
Constructors
+ RhymeSchemeSizeException(String) : void

gui Package

gui:: AutoPoet
Constructors
+ AutoPoet() : void
Methods
«synthetic» - lambda\$start\$(WindowEvent) : void + main(String[]) : void + start(Stage) : void

gui:: GetSuggestionsTask
Fields
«final» - filterParams : FilterParameters «final» - pos : PartOfSpeech «final» - suggestionParams : SuggestionPoolParameters «final» - superWord : SuperWord
Constructors
+ GetSuggestionsTask(SuperWord, PartOfSpeech, SuggestionPoolParameters, FilterParameters) : void
Methods
«volatile» «synthetic» «bridge» # call() : Object # call() : ArrayList<SuperWord>

gui:: IndexedTokenLabel
Constants
- SELECTABLE_CLASS : String - SELECTED_CLASS : String
Fields
- cntxtmnLabel : ContextMenu - controller : Controller - mnitmJoinWords : MenuItem - mnitmSplitOnHyphen : MenuItem - mnitmSplitOnSpace : MenuItem
Properties
«readOnly» + inclUnknown : boolean «readOnly» + lineIndex : int «readOnly» + poolParams : SuggestionPoolParameters + pos : PartOfSpeech «readOnly» + stanzaIndex : int + suggestions : ArrayList<SuperWord> «readOnly» + token : Token + tokenIndex : int

Constructors
+ IndexedTokenLabel(Controller, Token, int, int, int, boolean) : void
Methods
- highlightWord(IndexedTokenLabel) : void + incrementTokenIndex(int) : void + isNeighbour(IndexedTokenLabel) : boolean - onClick(MouseEvent) : void - populateContextMenu() : void - selectNeighbour() : void + setJoinWordsAction(EventHandler<ActionEvent>) : void + setSplitOnHyphenAction(EventHandler<ActionEvent>) : void + setSplitOnSpaceAction(EventHandler<ActionEvent>) : void + toString() : String - unhighlightWord(IndexedTokenLabel) : void

gui:: Controller
Constants
- SEE_LOG : String - SUGGESTION_CLASS : String
Fields
«@FXML» ~ anpnPoem : AnchorPane «@FXML» ~ anpnRoot : AnchorPane «@FXML» ~ btnGetSuggestions : Button «@FXML» ~ chbxInclUnknown : CheckBox ~ chbxSyllableCount : CheckBox «@FXML» ~ flwpnLine0 : FlowPane «@FXML» ~ flwpnSuggestions : FlowPane «@FXML» ~ grdPnFilters : GridPane «@FXML» ~ grdPnSuggestionPools : GridPane «@FXML» ~ grdpnPoem : GridPane «@FXML» ~ lblActRhymeScheme : Label «@FXML» ~ lblPoemLineCount : Label «@FXML» ~ lblPoemStanzaCount : Label «@FXML» ~ lblStanzaLineCount : Label «@FXML» ~ lblStanzaNumber : Label - poem : Poem - poemFile : File «@FXML» ~ rdbtnAdjective : RadioButton «@FXML» ~ rdbtnAdverb : RadioButton «@FXML» ~ rdbtnConjunction : RadioButton «@FXML» ~ rdbtnDefiniteArticle : RadioButton «@FXML» ~ rdbtnNoun : RadioButton «@FXML» ~ rdbtnPreposition : RadioButton

```

~ rdbtnPronoun : RadioButton
«@FXML»
~ rdbtnUnknown : RadioButton
«@FXML»
~ rdbtnVerb : RadioButton
~ rhymeTypeCheckBoxes : EnumMap<RhymeType, CheckBox>
«@FXML»
~ scrlnPoem : ScrollPane
«@FXML»
~ scrlnSuggestions : ScrollPane
~ suggestionPoolCheckBoxes : HashMap<String, CheckBox>
«@FXML»
~ tgbtnDirectEdit : ToggleButton
«@FXML»
~ tglgrpPoS : ToggleGroup
«@FXML»
~ tltplntRhymeScheme : Tooltip
«@FXML»
~ ttlpnCurrentStanza : TitledPane
«@FXML»
~ ttlpnPoem : TitledPane
«@FXML»
~ ttlpnSuggestionParameters : TitledPane
«@FXML»
~ ttlpnSuggestions : TitledPane
«@FXML»
~ txtarPoem : TextArea
«@FXML»
~ txtfldDefaultRhymeScheme : TextField
«@FXML»
~ txtfldIntRhymeScheme : TextField
«@FXML»
~ txtfldRhymeWith : TextField

```

Properties

```

«readOnly»
- filterParams : FilterParameters
+ focusedToken : IndexedTokenLabel
«readOnly»
- rhymesFromScheme : ArrayList<IndexedTokenLabel>
+ secondFocusedToken : IndexedTokenLabel
«readOnly»
+ suggestions : void

```

Constructors

```

+ Controller( ) : void

```

Methods

```

- addEmptyLine( int ) : FlowPane
- buildCleanAlert( AlertType ) : Alert
- buildEmptyLine( ) : FlowPane
- buildFilterCheckBox( String, String ) : CheckBox
- buildRhymeSchemeFormatter( ) : TextFormatter<String>
- buildRhymeTypeCheckBox( RhymeType ) : CheckBox
- buildSuggestionPoolCheckBox( SuggestionPool ) : CheckBox
- displaySuggestions( ) : void
- enableFilterBoxes( ) : void
- enableSuggestionPoolBoxes( ) : void
+ focusOnStanza( int ) : void
+ focusOnWord( IndexedTokenLabel ) : void
- getLastToken( FlowPane ) : IndexedTokenLabel
- getPoSRadioButton( PartOfSpeech ) : RadioButton
- initFilterCheckBoxes( ) : void
- initSuggestionPoolCheckBoxes( ) : void
«@FXML»
+ initialize( ) : void
- joinWords( ) : void
«synthetic»

```


- lambda\$buildRhymeSchemeFormatter\$3(Change) : Change
«synthetic»
- lambda\$displaySuggestions\$6(SuperWord, MouseEvent) : void
«synthetic»
- lambda\$getSuggestions\$4(IndexedTokenLabel, Task, WorkerStateEvent) : void
«synthetic»
- lambda\$initialize\$0(ActionEvent) : void
«synthetic»
- lambda\$initialize\$1(ActionEvent) : void
«synthetic»
- lambda\$initialize\$2(ActionEvent) : void
«synthetic»
- lambda\$newPoem\$5(String) : void
- makeSubstitution(SuperWord) : void
- + newPoem() : void
- + openPoem() : void
- + savePoem() : void
- + savePoemAs() : void
- splitWord(String) : boolean
- + toggleDirectEdit() : void
- tokenizePoem() : void
- updateDefaultRhymeScheme(TextField) : void
- + updateDefaultRhymeScheme(ActionEvent) : void
- + updateIncludeUnknowns(ActionEvent) : void
- + updateIntendedRhymeScheme(ActionEvent) : void
- + updatePartOfSpeech(ActionEvent) : void
- + updateSuggestionPool(ActionEvent) : void

utils Package

«memberClass» utils::ParameterWrappers:: FilterParameters
Fields
- matchWithPoS : PartOfSpeech - rhymeFilters : EnumMap<RhymeType, List<SuperWord>> - syllableCountFilter : boolean
Properties
+ matchPoS : PartOfSpeech
Constructors
+ FilterParameters() : void
Methods
+ getMatchWith(RhymeType) : List<SuperWord> «synthetic» - lambda\$setRhymeFilter\$0(RhymeType) : List + removeFilter(RhymeType) : void + setRhymeFilter(RhymeType, SuperWord) : void + setSyllableCountFilter(boolean) : void + syllableCountFilter() : boolean + toString() : String

«interface» utils:: ParameterWrappers

«final» «enum» «memberClass» utils::ParameterWrappers::FilterParameters:: RhymeType
Constants
«synthetic» - \$VALUES : RhymeType[] + FORCED_RHYME : RhymeType + IMPERFECT_RHYME : RhymeType + PERFECT_RHYME : RhymeType + SYLLABIC_RHYME : RhymeType + WEAK_RHYME : RhymeType
Properties
«final» «readOnly» + explanation : String «final» «readOnly» + label : String
Constructors
- RhymeType(String, int, String, String) : void
Methods
+ valueOf(String) : RhymeType + values() : RhymeType[]

utils:: NullListOperations
Constructors
+ NullListOperations() : void
Methods
+ addAllToNull(ArrayList<T>, Collection<T>) : ArrayList<T> + addToNull(ArrayList<T>, T) : ArrayList<T> + combineLists(Collection<ArrayList<T>>) : ArrayList<T> + combineListsPrioritiseDuplicates(ArrayList<ArrayList<T>>) : ArrayList<T> «transient» «@SafeVarargs» + combineListsVarargs(List<T>[]) : ArrayList<T>

utils:: Pair<O, T>
Fields
- one : O - two : T
Constructors
+ Pair(O, T) : void
Methods
+ one() : O + toString() : String + two() : T

utils:: LogWriter
Fields
- persistentLog : File - persistentWriter : FileWriter - tempLog : File - tempLogging : boolean - tempWriter : FileWriter
Constructors
+ LogWriter(boolean) : void
Methods
+ closeLogWriters() : void + main(String[]) : void + writePersistentLog(String) : void + writeTempLog(String) : void

«memberClass» utils::NullListOperations:: SortByCount<T>
Fields
- counts : Map<T, Integer>
Constructors
+ SortByCount(Map<T, Integer>) : void
Methods
+ compare(T, T) : int

«memberClass» utils::ParameterWrappers: SuggestionPoolParameters
Fields
- inclusiveUnknown : boolean - pools : EnumMap<SuggestionPool, Boolean>
Properties
«readOnly» + empty : boolean
Constructors
+ SuggestionPoolParameters() : void
Methods
+ hasInclusiveUnknown() : boolean + includes(SuggestionPool) : boolean + setInclusiveUnknown(boolean) : void + toString() : String + togglePool(SuggestionPool, boolean) : void

«final» «enum» «memberClass» utils::ParameterWrappers::SuggestionPoolParameters:: SuggestionPool
Constants
«synthetic» - \$VALUES : RhymeType[] + ANTONYMS : SuggestionPool + COMMONLY_TYPED : SuggestionPool + COMMON_CATEGORIES : SuggestionPool + HAS_CATEGORIES : SuggestionPool + HAS_PARTS : SuggestionPool + HAS_TYPES : SuggestionPool + IN_CATEGORY : SuggestionPool + PART_OF : SuggestionPool + SIMILAR_TO : SuggestionPool + SYNONYMS : SuggestionPool + TYPE_OF : SuggestionPool
Properties
«final» «readOnly» + apiProperty : boolean «final» «readOnly» + apiString : String «final» «readOnly» + label : String
Constructors
- SuggestionPool(String, int, String, String, boolean) : void
Methods
+ fromString(String) : SuggestionPool + valueOf(String) : SuggestionPool + values() : SuggestionPool[]