JavaScript is disabled on your browser.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/Feature.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/simplenlg/features/DiscourseFunction.html)
* [Next Class](http://docs.google.com/simplenlg/features/Form.html)
* [Frames](http://docs.google.com/index.html?simplenlg/features/Feature.html)
* [No Frames](http://docs.google.com/Feature.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#id.3znysh7)|
* Constr |
* [Method](#id.tyjcwt)
* Detail:
* [Field](#id.3dy6vkm)|
* Constr |
* Method

simplenlg.features

## Class Feature

* java.lang.Object
  + simplenlg.features.Feature
* public abstract class Feature  
  extends java.lang.Object
* This class defines a list of features which can be set up users of SimpleNLG. Note that there are three feature classes in SimpleNLG.
  + Feature: features typically set up developers invoking SimpleNLG
  + LexicalFeature: features typically set up the SimpleNLG lexicon
  + InternalFeature: features typically used internally by SimpleNLG
* Elements in the system can, in theory, take any kind of feature. Some features will only be expected by certain processors, however. Developers can define their own features but should choose names that do not conflict with those presented here.  
    
  The details for each feature are supplied in a table. The entries are:

|  |  |
| --- | --- |
| * **Feature name** | * This is the name that will appear in the element's feature list as produced by the toString() method or by calling getAllFeatureNames |
| * **Expected type** | * As features are represented as a Map connecting a String and an Object then, in theory, a feature can take any object as a value. This table entry defines the type that the SimpleNLG system expects. |
| * **Created by** | * Defines where the feature is created. In addition, all features can be added specifically by users |
| * **Used by** | * Defines which processors use the feature. |
| * **Applies to** | * Defines which structural, syntactical or lexical elements this feature is applied to. |
| * **Default** | * Any default values attributed to the feature are given here. |
|  |

Version: 4.0 Author: E. Reiter and D. Westwater, University of Aberdeen.

### Field SummaryFields

|  |  |
| --- | --- |
| * + Modifier and Type | * + Field and Description |
| * + static java.lang.String | * + [**ADJECTIVE\_ORDERING**](http://docs.google.com/simplenlg/features/Feature.html#ADJECTIVE_ORDERING) This feature determines if the adjectives should be reordered. |
| * + static java.lang.String | * + [**AGGREGATE\_AUXILIARY**](http://docs.google.com/simplenlg/features/Feature.html#AGGREGATE_AUXILIARY) This feature determines if the auxiliary verbs in a clause should be aggregated. |
| * + static java.lang.String | * + [**APPOSITIVE**](http://docs.google.com/simplenlg/features/Feature.html#APPOSITIVE) An appositive is a type of postmodifying phrase which is quasi-synonymous with, or a possible replacement of, the phrase it modifies. |
| * + static java.lang.String | * + [**COMPLEMENTISER**](http://docs.google.com/simplenlg/features/Feature.html#COMPLEMENTISER) The complementiser is the word that joins a subordinate clause to the parent clause. |
| * + static java.lang.String | * + [**CONJUNCTION**](http://docs.google.com/simplenlg/features/Feature.html#CONJUNCTION) This feature represents the word (or phrase) used for linking coordinated phrases together. |
| * + static java.lang.String | * + [**CONJUNCTION\_TYPE**](http://docs.google.com/simplenlg/features/Feature.html#CONJUNCTION_TYPE) This feature represents the type of conjunction this coordination represents. |
| * + static java.lang.String | * + [**CUE\_PHRASE**](http://docs.google.com/simplenlg/features/Feature.html#CUE_PHRASE) This feature represents the cue phrase of a sentence. |
| * + static java.lang.String | * + [**ELIDED**](http://docs.google.com/simplenlg/features/Feature.html#ELIDED) This features determines if the phrase is elided. |
| * + static java.lang.String | * + [**FORM**](http://docs.google.com/simplenlg/features/Feature.html#FORM) This feature dictates the form that a verb takes. |
| * + static java.lang.String | * + [**INTERROGATIVE\_TYPE**](http://docs.google.com/simplenlg/features/Feature.html#INTERROGATIVE_TYPE) This feature determines the type of interrogative (question) used for the clause. |
| * + static java.lang.String | * + [**IS\_COMPARATIVE**](http://docs.google.com/simplenlg/features/Feature.html#IS_COMPARATIVE) This flag determines if the Adjective or Adverb should be inflected into the comparative form. |
| * + static java.lang.String | * + [**IS\_SUPERLATIVE**](http://docs.google.com/simplenlg/features/Feature.html#IS_SUPERLATIVE) This flag determines if the Adjective or Adverb should be inflected into the superlative form. |
| * + static java.lang.String | * + [**MODAL**](http://docs.google.com/simplenlg/features/Feature.html#MODAL) The modal represents the modal auxiliary verb that is used in a verb phrase to express mood or tense. |
| * + static java.lang.String | * + [**NEGATED**](http://docs.google.com/simplenlg/features/Feature.html#NEGATED) The flag determines if the corresponding verb phrase should be negated. |
| * + static java.lang.String | * + [**NUMBER**](http://docs.google.com/simplenlg/features/Feature.html#NUMBER) This feature is used to determine if the element is to be represented in singular or plural form. |
| * + static java.lang.String | * + [**PARTICLE**](http://docs.google.com/simplenlg/features/Feature.html#PARTICLE) This feature represents a particle used in conjunction with a verb. |
| * + static java.lang.String | * + [**PASSIVE**](http://docs.google.com/simplenlg/features/Feature.html#PASSIVE) This flag shows if the phrase or clause should be written in the passive form. |
| * + static java.lang.String | * + [**PERFECT**](http://docs.google.com/simplenlg/features/Feature.html#PERFECT) This flag shows if the phrase or clause should be written in the perfect form. |
| * + static java.lang.String | * + [**PERSON**](http://docs.google.com/simplenlg/features/Feature.html#PERSON) This feature represents the first-person, second-person or third-person nature of the phrase. |
| * + static java.lang.String | * + [**POSSESSIVE**](http://docs.google.com/simplenlg/features/Feature.html#POSSESSIVE) This flag shows if the noun phrase should be written in the possessive form. |
| * + static java.lang.String | * + [**PROGRESSIVE**](http://docs.google.com/simplenlg/features/Feature.html#PROGRESSIVE) This flag determines if the verb phrase should be constructed in the progressive form. |
| * + static java.lang.String | * + [**PRONOMINAL**](http://docs.google.com/simplenlg/features/Feature.html#PRONOMINAL) This flag can be set for noun phrases where it is desirable to overwrite the subject with a suitable pronoun. |
| * + static java.lang.String | * + [**RAISE\_SPECIFIER**](http://docs.google.com/simplenlg/features/Feature.html#RAISE_SPECIFIER) This flag can be set when specifiers in a coordinated phrase should be raised. |
| * + static java.lang.String | * + [**SUPPRESS\_GENITIVE\_IN\_GERUND**](http://docs.google.com/simplenlg/features/Feature.html#SUPPRESS_GENITIVE_IN_GERUND) This flag is set when it is necessary to suppress the genitive when dealing with gerund forms. |
| * + static java.lang.String | * + [**SUPRESSED\_COMPLEMENTISER**](http://docs.google.com/simplenlg/features/Feature.html#SUPRESSED_COMPLEMENTISER) This flag determines if complementisers in subordinating clauses should be suppressed. |
| * + static java.lang.String | * + [**TENSE**](http://docs.google.com/simplenlg/features/Feature.html#TENSE) This flag represents the tense or temporal quality of a verb. |

### Method Summary

### Methods inherited from class java.lang.Objectequals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail

#### ADJECTIVE\_ORDERING public static final java.lang.String ADJECTIVE\_ORDERING This feature determines if the adjectives should be reordered. Some lexicons might support the positioning of adjectives and by default, this order will be used. The user can override the ordering of adjectives by setting this feature to false.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *adjectiveOrdering* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - Set by the phrase factory on noun phrases. It can be overwritten by the user. |
| * + - **Used by** | * + - The syntax processor will reorder adjectives in the premodifiers list if this feature has a value of true. |
| * + - **Applies to** | * + - Noun phrases only. |
| * + - **Default** | * + - Boolean.TRUE |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.ADJECTIVE_ORDERING)

#### AGGREGATE\_AUXILIARY public static final java.lang.String AGGREGATE\_AUXILIARY This feature determines if the auxiliary verbs in a clause should be aggregated.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *aggregateAuxiliary* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - Needs to be manually set by user. |
| * + - **Used by** | * + - The syntax processor will remove the auxiliary verbs from clauses when this is set to true. |
| * + - **Applies to** | * + - Clauses only. |
| * + - **Default** | * + - Boolean.FALSE |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.AGGREGATE_AUXILIARY)

#### COMPLEMENTISER public static java.lang.String COMPLEMENTISER The complementiser is the word that joins a subordinate clause to the parent clause. For example, the two clauses: *Timmy sang a song* and *moved Elizabeth to tears* can be joined with the complementiser *that* to form: *Timmy sang a song* ***that*** *moved Elizabeth to tears*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *complementiser* |
| * + - **Expected type** | * + - Can take any NLGElement or a string. It is recommended to use *InflectedWordElement as created by the PhraseFactory method createWord(...).* |
| * + - **Created by** | * + - The default is created by the PhraseFactory when creating clauses. |
| * + - **Used by** | * + - The syntax processor uses the value of this feature when adding complements to subordinate clauses. |
| * + - **Applies to** | * + - Clauses only. |
| * + - **Default** | * + - "that" represented by an InflectedWordElement if a lexicon is used, otherwise a StringElement. |

#### CONJUNCTION public static final java.lang.String CONJUNCTION This feature represents the word (or phrase) used for linking coordinated phrases together.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *conjunction* |
| * + - **Expected type** | * + - A String. |
| * + - **Created by** | * + - CoordinatedPhraseElement creates this feature automatically. The user can overwrite the value. |
| * + - **Used by** | * + - The syntax processor constructs the correct syntax for the coordinated phrases. |
| * + - **Applies to** | * + - CoordinatedPhraseElements. |
| * + - **Default** | * + - null |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.CONJUNCTION)

#### CONJUNCTION\_TYPE public static final java.lang.String CONJUNCTION\_TYPE This feature represents the type of conjunction this coordination represents.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *conjunctionType* |
| * + - **Expected type** | * + - ConjunctionType |
| * + - **Created by** | * + - CoordinatedPhraseElement sets this value automatically. It can be overwritten by the user. |
| * + - **Used by** | * + - The syntax processor constructs the correct syntax for the coordinated phrases. |
| * + - **Applies to** | * + - CoordinatedPhraseElements. |
| * + - **Default** | * + - ConjunctionType.COORDINATING |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.CONJUNCTION_TYPE)

#### APPOSITIVE public static final java.lang.String APPOSITIVE An appositive is a type of postmodifying phrase which is quasi-synonymous with, or a possible replacement of, the phrase it modifies. A typical example occurs with NPs, e.g.: *his house,* ***a large villa****, is on the hill* where the phrase *a large villa* is an appositive postmodifier of *his house*. Note that appositives are usually realised surrounded by commas. Accordingly, this feature is primarily used by the orthography processor to determine whether commas should be placed around a postmodifying phrase.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *appositive* |
| * + - **Expected type** | * + - boolean |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The orthography processor to determine comma placement. |
| * + - **Applies to** | * + - Any phrase which is a postmodifier of another phrase. |
| * + - **Default** | * + - false or null |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.APPOSITIVE)
    - CUE\_PHRASE  
      public static final java.lang.String CUE\_PHRASE  
      This feature represents the cue phrase of a sentence. Cue phrases sometimes appear at the start of sentences. In the following example, *however* forms the cue phrase:  
      ***However****, John played football instead.*

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *cuePhrase* |
| * + - **Expected type** | * + - NLGElement |
| * + - **Created by** | * + - Cue phrases need to be specifically added by the user. |
| * + - **Used by** | * + - The syntax processor places the cue phrase at the start of a sentence. |
| * + - **Applies to** | * + - Sentences. |
| * + - **Default** | * + - null |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.CUE_PHRASE)

#### ELIDED public static final java.lang.String ELIDED This features determines if the phrase is elided. Elided phrases are omitted from the final realisation.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isElided* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - The feature needs to be set by the user. |
| * + - **Used by** | * + - The syntax processor removes elided phrases. |
| * + - **Applies to** | * + - Noun phrases only. |
| * + - **Default** | * + - Boolean.FALSE |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.ELIDED)

#### FORM public static final java.lang.String FORM This feature dictates the form that a verb takes.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *form* |
| * + - **Expected type** | * + - Form |
| * + - **Created by** | * + - Initially created by the phrase factory when constructing verb phrases. The user is free to set the value accordingly. |
| * + - **Used by** | * + - The syntax processor set the form on sentences to be the same as that from the containing verb phrase. The syntax processor will also check a verb's form for the addition of supporting words such as *will* and *to*. The morphology processor uses the form to determine the actual inflection of the verb to be used. |
| * + - **Applies to** | * + - Nouns, verbs noun phrases and verb phrases. |
| * + - **Default** | * + - Form.NORMAL |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.FORM)

#### INTERROGATIVE\_TYPE public static final java.lang.String INTERROGATIVE\_TYPE This feature determines the type of interrogative (question) used for the clause.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *interrogativeType* |
| * + - **Expected type** | * + - InterrogativeType |
| * + - **Created by** | * + - The user must set this value. |
| * + - **Used by** | * + - The syntax processor uses this feature to correctly structure interrogative clauses. |
| * + - **Applies to** | * + - Clauses only. |
| * + - **Default** | * + - null. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.INTERROGATIVE_TYPE)

#### IS\_COMPARATIVE public static final java.lang.String IS\_COMPARATIVE This flag determines if the Adjective or Adverb should be inflected into the comparative form. For example, the comparative form for *early* is *earlier*, the comparative form of *big* is *bigger*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isComparative* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The morphology processor to correctly inflect the word. |
| * + - **Applies to** | * + - Adjectives and adverbs only |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.IS_COMPARATIVE)

#### IS\_SUPERLATIVE public static final java.lang.String IS\_SUPERLATIVE This flag determines if the Adjective or Adverb should be inflected into the superlative form. For example, the comparative form for *early* is *earliest*, the superlative form of *big* is *biggest*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isSuperlative* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The morphology processor to correctly inflect the word. |
| * + - **Applies to** | * + - Adjectives and adverbs only |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.IS_SUPERLATIVE)

#### MODAL public static final java.lang.String MODAL The modal represents the modal auxiliary verb that is used in a verb phrase to express mood or tense. The English modals are: *can*, *may*, *must*, *ought*, *shall*, *should*, *will* and *would*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *modal* |
| * + - **Expected type** | * + - String |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The syntax processor adds modals into the structure. |
| * + - **Applies to** | * + - Clauses, coordinated phrases and verb phrases. In the case of clauses and coordinated phrases, the modal is passed on to the underlying verb phrase. |
| * + - **Default** | * + - null. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.MODAL)

#### NEGATED public static final java.lang.String NEGATED The flag determines if the corresponding verb phrase should be negated. For example, negating the clause *John kissed Mary* results in the clause *John did not kiss Mary*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isNegated* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The syntax processor, which will correctly add in all necessary auxiliary verbs. |
| * + - **Applies to** | * + - Clauses, coordinated phrases and verb phrases. In the case of clauses and coordinated phrases, the modal is passed on to the underlying verb phrase. Applying negation to a coordinated phase will negate all the coordinates. |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.NEGATED)

#### NUMBER public static final java.lang.String NUMBER This feature is used to determine if the element is to be represented in singular or plural form.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *number* |
| * + - **Expected type** | * + - NumberAgreement |
| * + - **Created by** | * + - The phrase factory will create the number in certain cases, mostly in dealing with pronouns. However, with supporting lexicons a word such as *dogs* will be correctly identified as the plural form of *dog*. The user can also set this feature. |
| * + - **Used by** | * + - The syntax processor ensures number agreement across phrases and clauses, while the morphology processor uses the feature to pluralise words. |
| * + - **Applies to** | * + - Clauses, coordinated phrases, noun phrases and verb phrases. |
| * + - **Default** | * + - NumberAgreement.SINGULAR. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.NUMBER)

#### PARTICLE public static final java.lang.String PARTICLE This feature represents a particle used in conjunction with a verb. For example, the verb phrases *fall down* and *look up* have particles of *down* and *up* respectively.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *particle* |
| * + - **Expected type** | * + - String or NLGElement |
| * + - **Created by** | * + - The phrase factory will automatically check verbs to see if they contain more than one word. In such a case, the first word becomes the verb while additional words form the particle. The user can also explicitly specify a particle |
| * + - **Used by** | * + - The syntax processor adds particles into the structure. |
| * + - **Applies to** | * + - Verb phrases only. |
| * + - **Default** | * + - null. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.PARTICLE)

#### PASSIVE public static final java.lang.String PASSIVE This flag shows if the phrase or clause should be written in the passive form. For example, the clause *the cat ate the mouse* can be written in the passive form as *the mouse was eaten by the cat*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isPassive* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - User defined. |
| * + - **Used by** | * + - The syntax processor uses the passive feature to determine the correct ordering of subjects and objects in a clause. It also adds in additional auxiliary verbs to verb phrases, such as *was* in the above example. The morphology processor will change pronouns of noun phrases into passive form. |
| * + - **Applies to** | * + - Clauses, noun phrases and verb phrases. |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.PASSIVE)

#### PERFECT public static final java.lang.String PERFECT This flag shows if the phrase or clause should be written in the perfect form. The perfect aspect is normally formed from the auxiliary verb *to have* followed by the past participle of the main verb. For example, the phrase *the cat eats the mouse* would have the present perfect form of *the cat has eaten the mouse*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isPerfect* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - User defined. |
| * + - **Used by** | * + - The syntax processor adds in additional auxiliary verbs to verb phrases, and alters the form of the main verb. |
| * + - **Applies to** | * + - Verbs and verb phrases. |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.PERFECT)

#### PERSON public static final java.lang.String PERSON This feature represents the first-person, second-person or third-person nature of the phrase. This predominantly affects pronouns such as *I*, *you* and *they* but some verbs will also be modified depending on the person of reference. For example, kiss is used as the present tense for first and second person (*I kiss Mary* and *you kiss Mary*) while kisses is used for third person ( *he kisses Mary*).

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *person* |
| * + - **Expected type** | * + - Person |
| * + - **Created by** | * + - Will be set automatically by the phrase factory when personal pronouns are identified. May also be set by the user. |
| * + - **Used by** | * + - The syntax processor does some basic checking on the person-nature of phrases while the morphology processor will correctly inflect pronouns and verbs. |
| * + - **Applies to** | * + - Clauses, coordinated phrases, noun phrases and verb phrases. |
| * + - **Default** | * + - Person.THIRD. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.PERSON)

#### POSSESSIVE public static final java.lang.String POSSESSIVE This flag shows if the noun phrase should be written in the possessive form. The possessive form of a noun is usually formed from the noun with *'s* added to the end. For example, *dog* has a possessive form *dog's* . Certain personal pronouns follow different rules, the possessive form of *I* is *mine*, *you* is *yours*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isPossessive* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - User defined. |
| * + - **Used by** | * + - The syntax processor correctly defines the possessive on noun phrases that contain more than one subject (we say *John and Mary's* not *John's and Mary's* ). The morphology processor correctly inflects the possessive forms of nouns and pronouns. |
| * + - **Applies to** | * + - Clauses, coordinated phrases and noun phrases. |
| * + - **Default** | * + - Boolean.FALSE in most cases or for pronouns it is dependent on the pronoun form used. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.POSSESSIVE)

#### PRONOMINAL public static final java.lang.String PRONOMINAL This flag can be set for noun phrases where it is desirable to overwrite the subject with a suitable pronoun.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isPronominal* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - User defined. |
| * + - **Used by** | * + - The syntax processor will replace the subject of a noun phrase with a suitable personal pronoun. |
| * + - **Applies to** | * + - Noun phrases only. |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.PRONOMINAL)

#### PROGRESSIVE public static final java.lang.String PROGRESSIVE This flag determines if the verb phrase should be constructed in the progressive form. For example, the progressive form of *John kisses Mary* is *John is kissing Mary*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *progressive* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - Set by the user. |
| * + - **Used by** | * + - The syntax processor adds in required auxiliary verbs when dealing with the progressive form. |
| * + - **Applies to** | * + - Clauses, coordinated phrases and verb phrases only. |
| * + - **Default** | * + - Boolean.FALSE. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.PROGRESSIVE)

#### RAISE\_SPECIFIER public static final java.lang.String RAISE\_SPECIFIER This flag can be set when specifiers in a coordinated phrase should be raised. For example, the coordinated phrase *my cat and my dog* can have its specifiers raised becoming *my cat and dog*.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *raiseSpecifier* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - User defined. |
| * + - **Used by** | * + - The syntax processor to correctly add or remove specifiers. |
| * + - **Applies to** | * + - Coordinated phrases only. |
| * + - **Default** | * + - Boolean.FALSE |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.RAISE_SPECIFIER)

#### SUPPRESS\_GENITIVE\_IN\_GERUND public static final java.lang.String SUPPRESS\_GENITIVE\_IN\_GERUND This flag is set when it is necessary to suppress the genitive when dealing with gerund forms.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *suppressGenitive* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The syntax processor will set subjects to be possessive when dealing with the gerund form unless this flag is set. |
| * + - **Applies to** | * + - Clauses only. |
| * + - **Default** | * + - Boolean.FALSE |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.SUPPRESS_GENITIVE_IN_GERUND)

#### SUPRESSED\_COMPLEMENTISER public static final java.lang.String SUPRESSED\_COMPLEMENTISER This flag determines if complementisers in subordinating clauses should be suppressed.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *isSuppressedComplementiser* |
| * + - **Expected type** | * + - Boolean |
| * + - **Created by** | * + - The phrase factory sets some defaults but can be overridden by the user. |
| * + - **Used by** | * + - The syntax processor will ignore suppressed complementisers on subordinating clauses. |
| * + - **Applies to** | * + - Clauses and coordinated phrases. |
| * + - **Default** | * + - Boolean.FALSE for clauses or Boolean.TRUE on clauses when added to a coordinated phrase. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.SUPRESSED_COMPLEMENTISER)

#### TENSE public static final java.lang.String TENSE This flag represents the tense or temporal quality of a verb.

|  |  |
| --- | --- |
| * + - **Feature name** | * + - *tense* |
| * + - **Expected type** | * + - Tense |
| * + - **Created by** | * + - The user. |
| * + - **Used by** | * + - The syntax processor ensures tense on phrases, clauses and coordinated phrases is passed on to the underlying verb phrases. The morphology processor uses the tese to correctly inflect verbs. |
| * + - **Applies to** | * + - Clauses, coordinated phrases and verb phrases. |
| * + - **Default** | * + - Tense.PRESENT. |

* + - See Also:[Constant Field Values](http://docs.google.com/constant-values.html#simplenlg.features.Feature.TENSE)

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/Feature.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/simplenlg/features/DiscourseFunction.html)
* [Next Class](http://docs.google.com/simplenlg/features/Form.html)
* [Frames](http://docs.google.com/index.html?simplenlg/features/Feature.html)
* [No Frames](http://docs.google.com/Feature.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#id.3znysh7)|
* Constr |
* [Method](#id.tyjcwt)
* Detail:
* [Field](#id.3dy6vkm)|
* Constr |
* Method