Executive Summary

This program will be a very basic chatbot. Essentially it will simulate very basic Artificial Intelligence (AI) through interaction with the user. The program will accept input in the form of grammatically correct English and respond appropriately—also in English.

The program will utilize dictionary files to detect words and determine the part of speech they belong to. (i.e. noun, verb, article, etc.) It will then attempt to construct a comprehensible sentence by analyzing how those parts of speech should interact based on a set of rules.

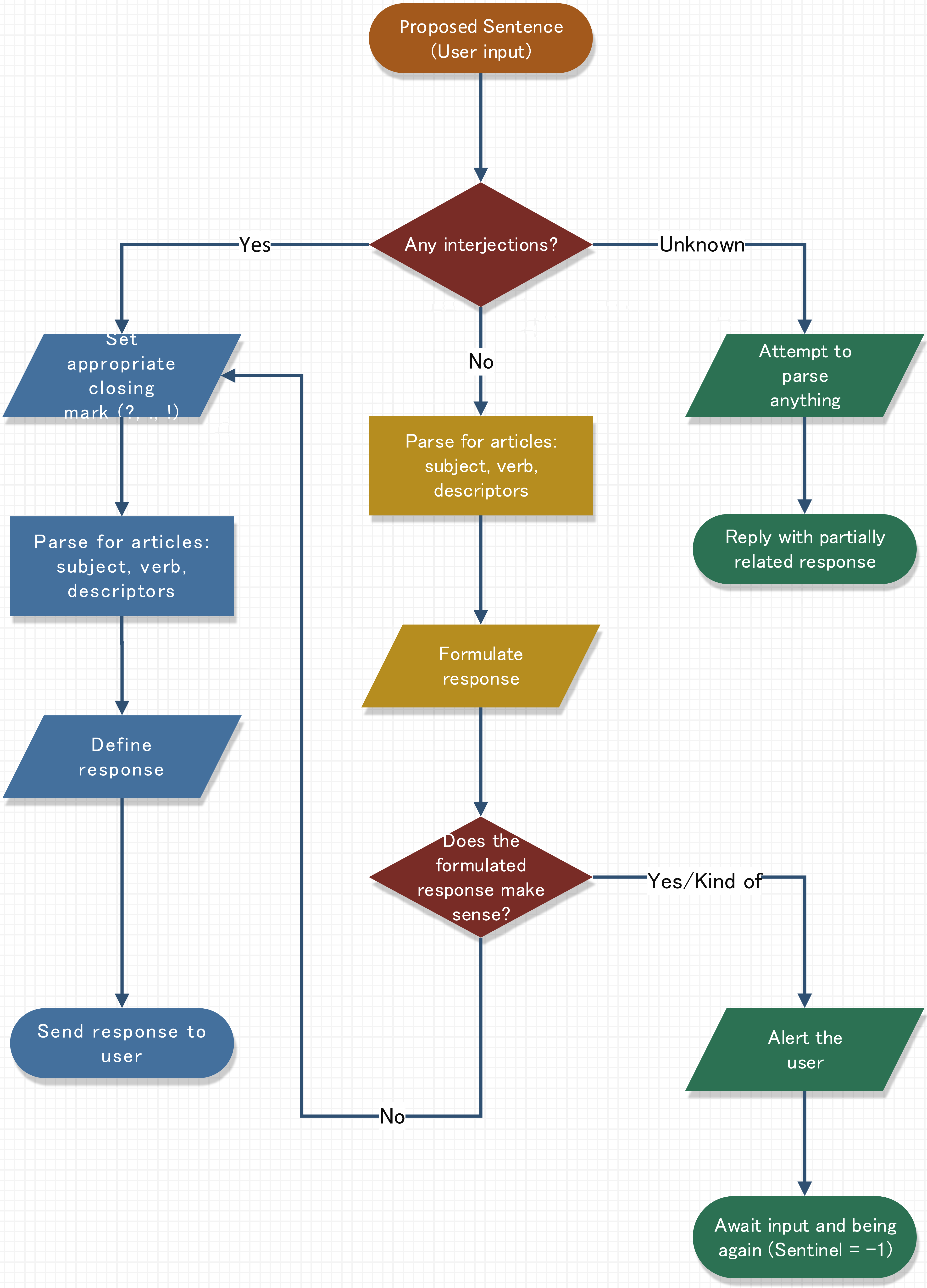
The program will then attempt to construct an appropriate response to the user's input. It will use the words received from the user as well as words in its dictionary files to formulate a response in the form of a statement or question related to the user input.

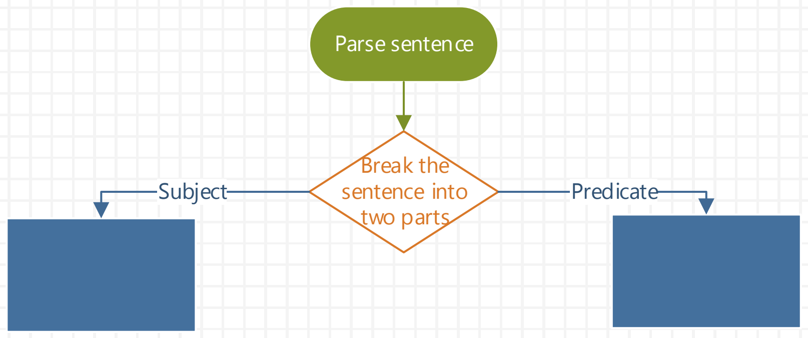
The program will not be required to handle incorrect grammar, slang, profanity, obscure vocabulary or non-English words. Once the minimum requirements are met adding that functionality is optional.

Business Requirements

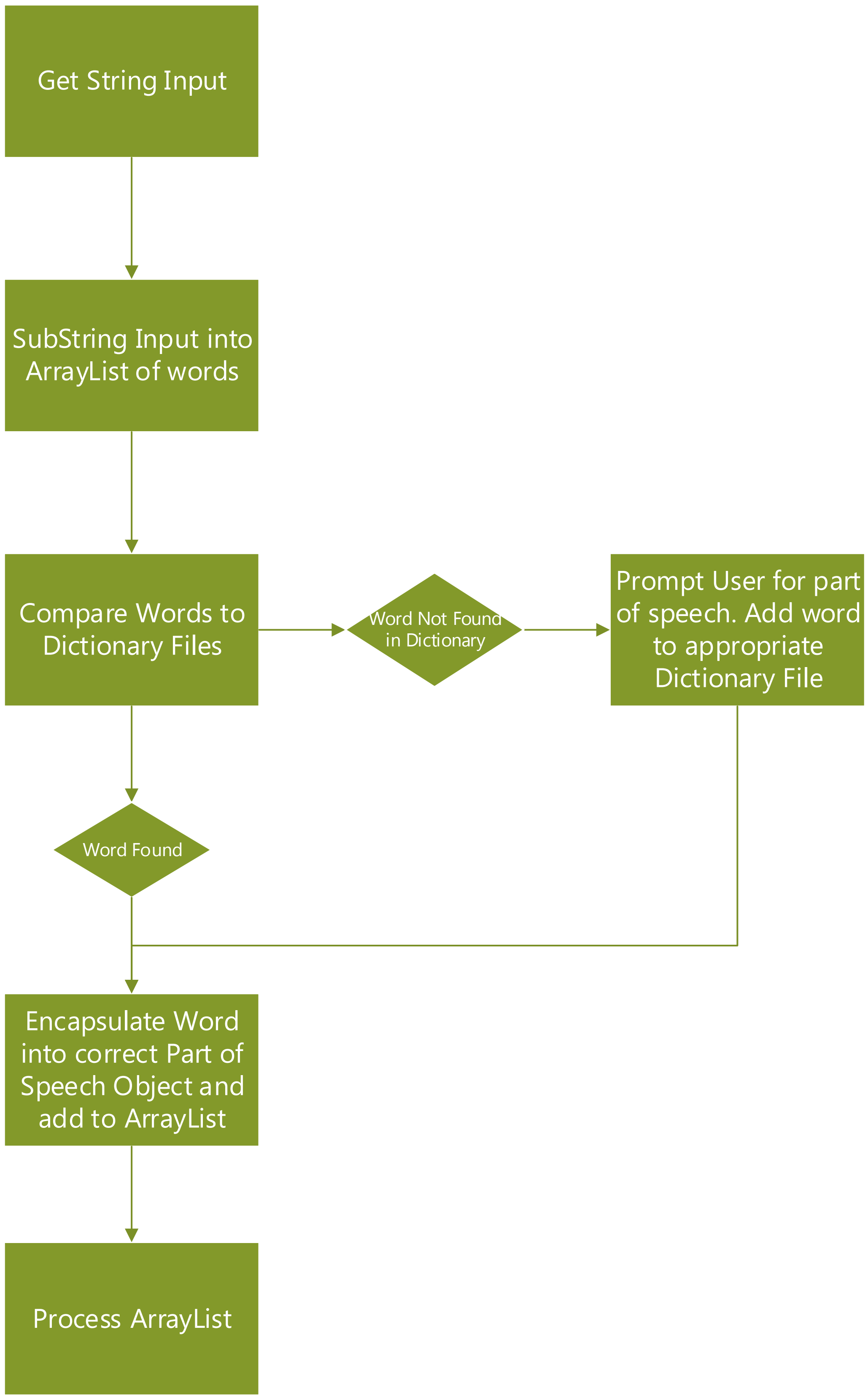
- Required
    - Must accept a sentence from the user
    - Must respond with a relevant statement or question
    - Must at least interact through the console
- Stretch
    - Can store and "remember" new words it learns through interaction with the user
    - Can detect and decipher part of speech an unknown word belongs to based on context
    - Can interact through a basic Graphical Interface (GUI)
    - Can detect and correct errors in spelling and grammar

Language Processor Program Flow


Proposed Sentence (User input)
Any interjections?
Yes
No
Unknown
Set appropriate closing mark (?, ., !)
Parse for articles: subject, verb, descriptors
Define response
Send response to user
Parse for articles: subject, verb, descriptors
Formulate response
Does the formulated response make sense?
No
Yes/Kind of
Attempt to parse anything
Reply with partially related response
Alert the user
Await input and being again (Sentinel = -1)


Parse sentence
Break the sentence into two parts
Subject
Predicate

Input Parsing


Get String Input
SubString Input into ArrayList of words
Compare Words to Dictionary Files
Word Not Found in Dictionary
Prompt User for part of speech. Add word to appropriate Dictionary File
Word Found
Encapsulate Word into correct Part of Speech Object and add to ArrayList
Process ArrayList

Chat Bot Interaction

Interactions with the Chat Bot will consist of three main stages, input parsing, input interpretation, and response. The input parsing stage involves obtaining user input, validating it, and then organizing it into a data structure that is ready to be interpreted. Input interpretation involves processing the input through a series of grammatical rules to determine if the input forms a valid sentence. The response stage involves constructing a valid response to the input. The response will be a simple statement or question, potentially guiding further conversation through prompts to the user or, if the chat bot has failed to understand the input, a change of subject. The three stages are explained in more detail below.

Input Parsing

The process of parsing the user's input begins with obtaining the user's input through the console. This input will be stored as a String. This input string will then be cut into substrings at each space in order to obtain individual words. During this process these strings are collected and stored in an ArrayList of strings. Next each word will then be checked against the dictionary files in order to determine its classification as a part of speech. If the word is found in a dictionary it is then wrapped in the appropriate part of speech object that is used during interpretation. In the event a word is not found in the dictionary the user will be prompted to confirm that the word is a valid word and then provide the correct classification. That word is then added to the appropriate dictionary and wrapped in the appropriate object. This process allows the chat bot to "learn" more words as it is used. During this process the part of speech objects are collected in an ArrayList of part of speech objects.

Interpretation

The interpretation of the input occurs on a stack and is dictated by grammatical rules which are expressed as an algorithm. Basically, we are concerned with articles, nouns, and verbs which can be combined in various ways to form noun phrases and verb phrases which form sentences. Using the ArrayList of part of speech objects our algorithm will push each object onto the stack, one at a time, and determine with each push if what is on the stack can be "reduced" into a more general part of a sentence. For example, the sentence, "The dog runs" which will have already been parsed in the appropriate objects, will be interpreted in the following manner:

- Article("The") is pushed onto the stack.
- The stack contains only an Article which cannot be reduced.
- Noun("Dog") is pushed onto the stack.
- The stack contains an Article and a Noun which can be reduced to a Noun Phrase.
- Article("The") and Noun("Dog") are popped off the stack and reduced to NounPhrase(Article("The"), Noun("Dog")) which is pushed onto the stack.
- The stack contains only a Noun Phrase which cannot be reduced.
- Verb("runs") is pushed onto the stack.
- The stack contains a Noun Phrase and a Verb which can be reduced to a Noun Phrase and a Verb Phrase.
- Verb("runs") is popped of the stack and reduced to VerbPhrase("runs") which is pushed back onto the stack.
- The stack contains a Noun Phrase and a Verb Phrases which can be reduced to a sentence.
- There are no more objects to push so the Noun Phrase and Verb Phrase are popped off the stack and reduced to a Sentence object.

In the event the input objects cannot be reduced the chat bot will prompt the user that it was unable to understand the input and prompt for new input. The resulting sentence object is stored for use in the response stage.

Response

The Response stage will make use of the valid sentence object to generate responses to the user's input. The responses will consist of predesigned response templates which will can be dynamically altered to produce relevant questions and statements. For example:

Given the input "My dog runs" The chat bot may respond with a question template: "Why does (pronoun) (noun) (verb)?" The parts of speech are dynamically filled in and the output to the user is, “Why does your dog run?” The user may then respond with more input. The chat bot will attempt to continue a conversation until the language becomes too complex at which time it will attempt to change the subject.

NLP::WordProcessor(Don't sue)
sentrofizzle() : ArrayList<String>
NLP::Controller
words : wordStack
uf : UserInnerface
processor : WordProcessor
+run() : void
-calcResponse() : void
-dicSearch(ArrayList<String>) : v
+main(String[] args) : void
NLP::UserInnerface
welcome() : void
mainMenu() : int
chat() : String
respond(String response) : void
Stack
«extends»
NLP::WordStack
setenceParser() : void
NLP::PartOfSpeech
«abstract»
FileIO
adverb: File
adjective: File
interjection: File
interrogatives: File
preposition: File
article: File
verb : File
helperVerb: File
noun : File
pronoun: File
punctuation: File
writeTo(File current) : void
surtch(File current) : boolean
«extends»
«extends»
«extends»
NLP::Word
«Abstract»
-word : String
Word(String word)
+getWord() : String
NLP::Punctuation
-punc : String
+getWord() : String
NLP::Phrase
«Abstract»
phrase : ArrayList<Word>
getPhrase : ArrayList<Word>
«extends»
«extends»
«extends»
«extends»
«extends»
«extends»
«extends»
«extends»
«extends»
«extends»
«extends»
NLP::Pronoun
-word : String
Pronoun(String word)
+getWord() : String
NLP::Noun
-word : String
Noun(String word)
+getWord() : String
NLP::Verb
-word : String
Verb(String word)
+getWord() : String
NLP::Article
-word : String
Article(String word)
+getWord() : String
NLP::Preposition
-word : String
Preposition(String word)
+getWord() : String
NLP::Interrogative
-word : String
Interrogative(String word)
+getWord() : String
NLP::Interjection
-word : String
Interjection(String word)
+getWord() : String
NLP::Adjective
-word : String
Adjective(String word)
+getWord() : String
NLP::Adverb
-word : String
Adverb(String word)
+getWord() : String
NLP::VerbPhrase
phrase : ArrayList<Word>
getPhrase : ArrrayList<Word>
NLP::NounPhrase
phrase : ArrayList<Word
getPhrase : ArrayList<Word
«extends»
NLP::HelperVerb
-word : String
+getWord() : String