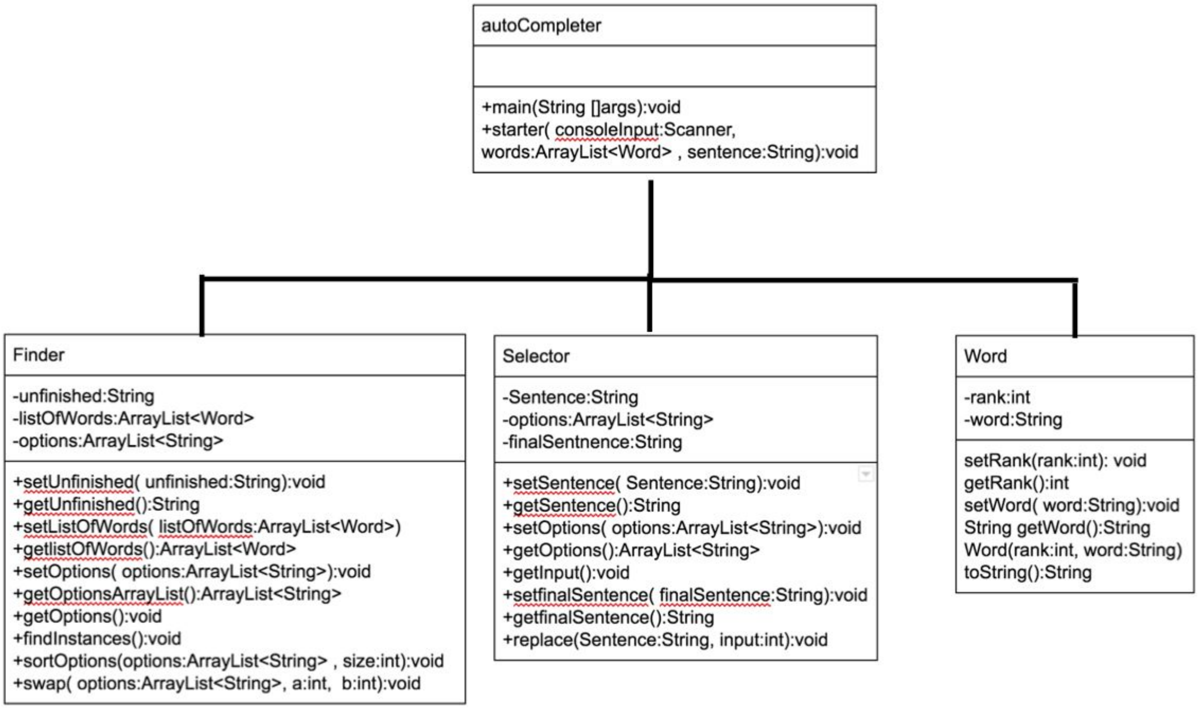
CSE 205: 17566 / M W 4:35-5:50
Assignment: Assignment #6
Author(s): Humdaan Mustafa & 10042025
Description: This Document contains the UML diagram used for this program as well as an analysis on the Big-O Notation for a method and a comparison to another method.

UML


autoCompleter
+main(String []args):void
+starter(consoleInput:Scanner, words:ArrayList<Word> , sentence:String):void
Finder
-unfinished:String
-listOfWords:ArrayList<Word>
-options:ArrayList<String>
+setUnfinished(unfinished:String):void
+getUnfinished():String
+setListOfWords(listOfWords:ArrayList<Word>)
+getlistOfWords():ArrayList<Word>
+setOptions(options:ArrayList<String>):void
+getOptionsArrayList():ArrayList<String>
+getOptions():void
+findInstances():void
+sortOptions(options:ArrayList<String> , size:int):void
+swap(options:ArrayList<String>, a:int, b:int):void
Selector
-Sentence:String
-options:ArrayList<String>
-finalSentnence:String
+setSentence(Sentence:String):void
+getSentence():String
+setOptions(options:ArrayList<String>):void
+getOptions():ArrayList<String>
+getInput():void
+setfinalSentence(finalSentence:String):void
+getfinalSentence():String
+replace(Sentence:String, input:int):void
Word
-rank:int
-word:String
setRank(rank:int): void
getRank():int
setWord(word:String):void
String getWord():String
Word(rank:int, word:String)
toString():String

Big-O Notation:
The time complexity of the findInstances() method found in the Finder.java file is O(N) where N is the size of the listOfWords. The list of words is close to 10,000 words and the for loop tests each and every word contained in the list.

The second method that could be used to find instances would be to use a binary search, assuming that the list is sorted alphabetically, which in this case it is not but could be. A regular binary search would be of O(logN). But in this case because we are not looking for one value, but all the instances of the unfinished word, there would be an additional complexity in finding all the instances. This additional function would add an O(N) time complexity to the algorithm, as we would have to find N additional instances of the word, whether they be greater than or less than the word found by the binary search. N would be relatively small in majority of cases. Because of Big O addition rules this would still equal O(logN).