CS 1440 Group Project – Vocabulary Quiz Game

In this group assignment, you will work with 2-3 classmates (group size 3-4) to build a game to help study the concepts we have learned in CS 1440 this semester. Below you are provided with UML diagrams of three of the required classes along with descriptions of the class members in each class. You will also write a driver program according to the specifications in this document. On the course page you will find a Vocabulary.txt file. Each group member must select 20 different words from this list and define them (you may not choose the same words as another group member). Choosing words that you do not know the definition of will help you study later. Instructions are provided for how to format the text file to hold the terms and definitions.

Materials Provided:

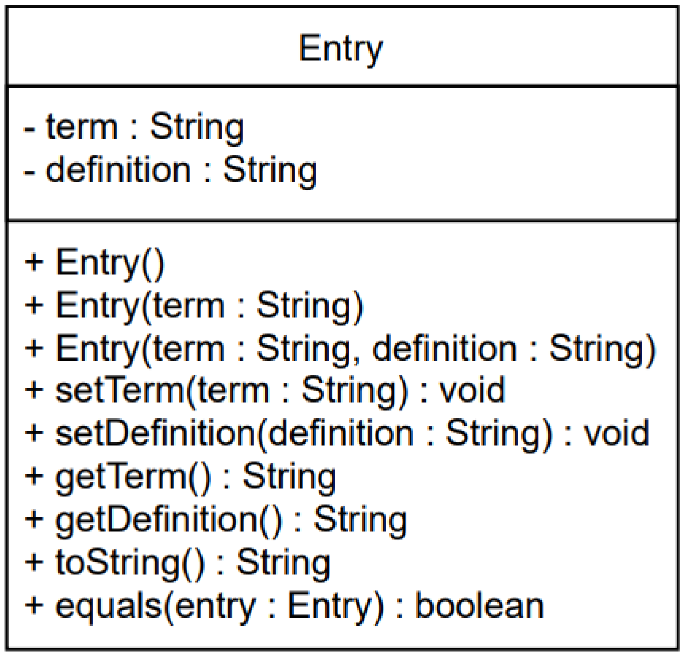
- Skeleton code for Entry.java, Glossary.java, GamePlay.java, and VocabQuiz.java
- List of vocabulary words to choose from (20 different terms per group member) in Vocabulary.txt
- UML diagrams for the Entry, Glossary, and GamePlay classes
- Documentation for the Entry, Glossary, GamePlay, and VocabQuiz classes
- Text file to store terms and their definitions in required format

Rubric

Criteria	10	7	5	3	0
20 Vocab Words per group member	Completed	Missing a small fraction of entries	Missing half of expected entries	Missing large fraction of entries	Not completed
Entry class written correctly - follows UML and specifications	Completed	Some methods do not behave as specified	Half of the methods do not behave as specified	Most of the methods do not behave as specified	Not completed
Glossary class written correctly - follows UML and specifications	Completed	Some methods do not behave as specified	Half of the methods do not behave as specified	Most of the methods do not behave as specified	Not completed
GamePlay class written correctly - follows UML and specifications	Completed	Some methods do not behave as specified	Half of the methods do not behave as specified	Most of the methods do not behave as specified	Not completed
VocabQuiz class written correctly - follows specifications	Completed	Some tasks were not completed as specified	Half of the tasks were not completed as specified	Most of the tasks were not completed as specified	Not completed
VocabList.txt formatted correctly	Completed	Some entries not formatted as specified	Half of the entries not formatted as specified	Most of the entries not formatted as specified	Not completed
Definitions are correct	Completed	Some entries are not properly defined	Half of the entries are not properly defined	Most of the entries are not properly defined	Not completed
Coding style is acceptable - easily readable - follows Checkstyle	Completed	Some coding style and comments are not readable	Half of the coding style and comments are not readable	Most of the coding style and comments are not readable	Not completed
Program compiles	Completed	One of the classes does not compile	Half of the classes do not compile	Most of the classes do not compile	Not completed
Program works as expected - game can be played to specifications	Completed	Some of the specified components of game play do not work as specified	Half of the specified components of game play do not work as specified	Most of the specified components of game play do not work as specified	Not completed

Entry Class

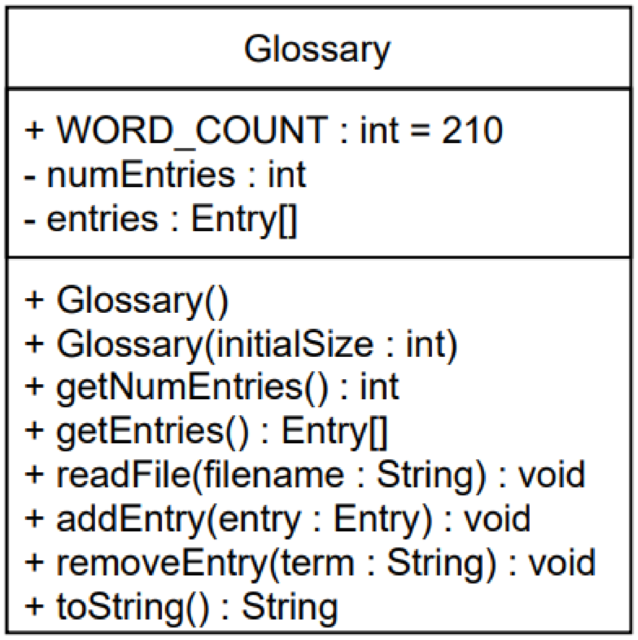
Entry.java is the class that we will make glossary entries from for the Vocabulary Quiz Game. It should follow the specifications below the UML diagram.


Entry
- term : String
- definition : String
+ Entry()
+ Entry(term : String)
+ Entry(term : String, definition : String)
+ setTerm(term : String) : void
+ setDefinition(definition : String) : void
+ getTerm() : String
+ getDefinition() : String
+ toString() : String
+ equals(entry : Entry) : boolean

- No-argument constructor that sets the term and definition fields to values that prevent Null Pointer Exceptions
- One-argument constructor that sets the term field to the parameter value and sets the definition field to a value that prevents a Null Pointer Exceptions
- Two-argument constructor that sets the term and definition fields to the parameter values
- Mutator and accessor methods for the two fields
- The toString method that returns a string with the following format - "%S - %s.\n"
- The equals method that returns true if the calling object and the parameter object have the same term and definition

Glossary Class

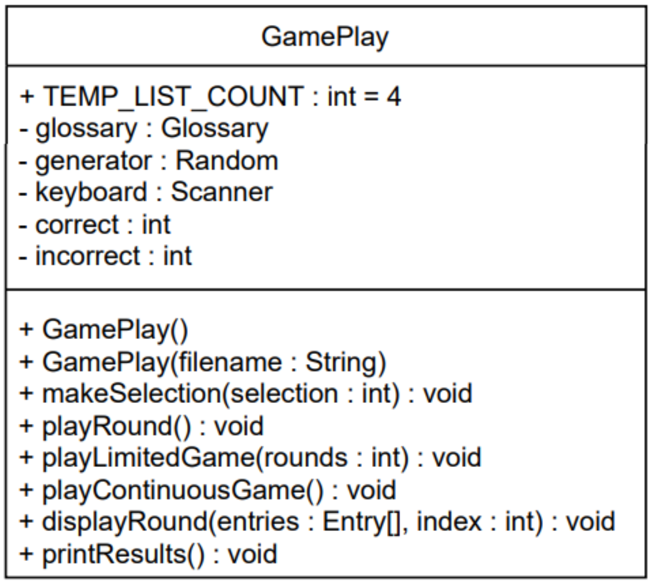
Glossary.java is the class that will hold an array of Entry items for the Vocabulary Quiz Game. It should follow the specifications below the UML diagram.



- No-argument constructor that sets the numEntries field to zero and the entries field to a new, single dimension array of type Entry using the constant as a size declarator
- One-argument constructor that sets the numEntries field to zero and the entries field to a new, single dimension array of type Entry using the parameter value as a size declarator
- Accessor methods for the numEntries and entries fields
- The readFile method that takes a String parameter for the filename needed for reading input. This method opens the file for reading and while there is still something left to read, we read a single line into a String object. The String class's split method must be used to create an array of Strings that we will use to make a new Entry object to add to the Glossary. Don't forget to close the Scanner when you are done reading the file.
- The addEntry method accepts an Entry object as an argument and places that Entry object at the next available index in the array of entries that is maintained by the numEntries field. The total number of entries must also be incremented.
- The removeEntry method accepts a String object as an argument for the term that we wish to remove from the Glossary. We must traverse the array until we find the term specified in the parameter and then we must set that index in the Glossary to null. The last task for this method is to decrement the number of entries in the Glossary.
- The toString method returns a String object that contains every Entry in the Glossary on a newline. We must be sure that we only print the indices that are not null from the Glossary.

GamePlay Class

GamePlay.java is the class that contains the methods for playing the Vocabulary Quiz Game. It should follow the specifications below the UML diagram.



```
GamePlay
+ TEMP_LIST_COUNT : int = 4
- glossary : Glossary
- generator : Random
- keyboard : Scanner
- correct : int
- incorrect : int
+ GamePlay()
+ GamePlay(filename : String)
+ makeSelection(selection : int) : void
+ playRound() : void
+ playLimitedGame(rounds : int) : void
+ playContinuousGame() : void
+ displayRound(entries : Entry[], index : int) : void
+ printResults() : void
```

- No-argument constructor that sets the glossary field a new Glossary object, sets the generator field to a new Random object, sets the keyboard field to a new Scanner object to read from standard input, uses the glossary field to call the readFile method using the VocabList.txt file as an argument, and sets the correct and incorrect fields to zero.
- One-argument constructor that sets the glossary field a new Glossary object, sets the generator field to a new Random object, sets the keyboard field to a new Scanner object to read from standard input, uses the glossary field to call the readFile method using theparameter value as an argument, and sets the correct and incorrect fields to zero.
- The makeSelection method MUST use a **switch statement** to determine what methods are called to play the Vocabulary Quiz Game. Switch on the parameter value and call the correct method(s) to perform the operations written in the menu options displayed in the VocabQuiz class. If the user chooses to play a Short Round, then 5 rounds will be played. If the user chooses to play a Long Round, then 20 rounds will be played. If the user chooses to exit the game, print "See ya next time" to the terminal. Don't forget the default case.
- The playRound method is responsible for picking a term at random from the Glossary that we want the user to define, picking at random a specified number of incorrect definitions to use as options in each round, calling the displayRound method, determining whether the user selected the correct definition, and incrementing the correct field for the result of that round. This method must create an array of Entry objects to send to the displayRound method along with the index at which the correct definition for the term randomly selected can be found in the array. The term is selected at random from an index in the Glossary and the index to store the randomly selected entry in the Entry array being passed to displayRound is also selected at random, so that the correct answer is not always in the same place. This method is one of the most difficult methods in the project and we may develop the code for this together in class.
- The playLimitedGame method accepts an argument for how many rounds the user wishes to play and calls the playRound method that number of times.
- The playContinuousGame method does not accept any arguments. This method calls the playRound method while the user wishes to continue. You must prompt the user after each round to determine if play should continue. The prompt should say "Would you like to play another round (enter 0 to continue or -1 to quit)? followed by a tab. You MUST use a **while loop** in this method.
- The displayRound method accepts an Entry array and integer as arguments. It tells the user to "Select the correct definition for the following term: " on its own line. Then the method prints the term for which we want the user to select a definition. The array of entries is traversed and the definition for each Entry object is printed after a number and a space starting with the number one.
- The printResults method prints the number of correct and incorrect answers with labels for each on its own line. You MUST use a **printf statement** in this method that provides a field width of 8 places, left aligns the data, uses a tab to separate the results, and the labels should be "Correct" and "Incorrect" respectively.

VocabQuiz Class

VocabQuiz.java is the Driver class for the Vocabulary Quiz Game. It should follow the specifications below.

- This class should contain only a main method
- You will need to use a Scanner object to read input from the keyboard
- You MUST use a **do-while loop** to prompt the user to select an option from the menu
- The prompt and menu should look exactly like this:
  Select a number from the menu below.
  1 - Print All Entries
  2 - Play Short Round
  3 - Play Long Round
  4 - Continuous Play
  5 – EXIT
- You are to make a **GamePlay** object and pass the filename "**VocabList.txt"** as the argument one-argument constructor.
- You are to call **makeSelection** method and pass the user's input as an argument

VocabList.txt

The text file that stores the terms and their definitions must be formatted as follows.

term: definition.
term: definition.
term: definition.
term: definition.
-
-
-

term: definition.

This formatting allows for us to use the Scanner class's nextLine method and the String class's split method.

After you have finished the entire project and submitted a zip file to AsULearn, your final task is to play your game to help you study for the final. If you have any questions or concerns, please ask in class as we work on the project or ask your group mates. Emails are always welcome as well. This project will be graded out of a total of 100 points according to the rubric found at the beginning of this document. This project is worth 15% of your final lecture exam. Consideration will be taken for contribution to the project if any complaints from group members are raised. Please be a team player. If your group decides to split the project up into pieces and assign tasks to group members, then records of who did what will be required. For example, if your group splits up the work by methods, then a comment needs to be added to that method with the group member's name that wrote the method or if the project is split up by class, then the author tags need to reflect the group member that completed that class. The splitting up of work is up to you and what works best for your team.