Andrew Halisky

CSE 210: Programming with Classes

Brother Poulson

May 27, 2023

For our project this week, our team came up with different names for class attributes for the Scripture Memorizer Project. This project will use a specific code to hide different words in a scripture every time the user presses enter. It will do this until the scripture is completely covered in underscores representing each word in the scripture to memorize. Then a new scripture will generate and it will repeat the same process again. When the user types “quit” it will exit the program completely.

The program will be written using the following class attributes:

* In Program.cs
  + In the main program, it will start off with a Console.WriteLine() for two lines that introduce the user to the program, telling them to press enter to generate a scripture or type “quit” to leave the program.
  + There will be a bool exit that will equal false at the top with a while loop running the program with (!exit). When the user types “quit,” the exit will equal true, and a Console.WriteLine() will be written saying goodbye and then a break will be inserted down below.
  + There will be a for loop that will allow multiple words to be hidden with an underscore 10 times per scripture. Then there will be references to two methods in the Scripture.cs file to display each scripture properly.
* In Word.cs
  + Will add the public class Word behavior:
    - Will add two Lists:
      * The first List will be entitled \_hiddenwords which will store the hidden words in a list.
      * The second List will be entitled \_shownwords which will store the list of words that were shown in the program in a list
    - Will add a string constructor Hide() to add and return the hidden words from the \_hiddenwords list.
    - Will add a string constructor called Show() to add and return the words that were shown from the \_shownwords list.
    - The constructor IsHidden() void will be used to store the words that were hidden every time the enter key was pressed.
    - The constructor GetRenderedText() void will contain a code that will be used to convert different words to an underscore.
* In ScriptureGenerator.cs
  + Will add the public class ScriptureGenerator behavior:
    - Will add two more Lists:
      * The first List will be entitled \_words which will contain a string list of all of the sentences of scriptures separated by a comma.
      * The second List will be entitled \_references which will contain a string list of the references of each of those scriptures including the books, chapters, and verses.
    - Will contain a GenerateScripture() constructor string that will get a random generated scripture and return it.
* In References.cs
  + Will add the public class References behavior:
    - Will include a \_chapter int method which will get the chapter index from the text in the \_references list.
    - Will include a \_book string method which will get the book index from the text in the \_references list.
    - Will include a \_startVerse int method which will get the index of the start of the verse in the \_words list.
    - Will include a \_endVerse int method which will get the index of the end of the verse in the \_words list.
    - The FormatReference() string constructor will contain a code that will format each word that is hidden every time a user presses enter.
    - The Display Format string constructor will call the Show() constructor method and include a foreach loop that will get each word in the text and return it.
* In Scriptures.cs
  + Will add the public class Scripture behavior:
    - Will include a HideWords() void constructor that will be used call the IsHidden() method where the hidden words were stored.
    - Will include a GetRenderedText() constructor which will be used to call the method of the same name from the Word.cs file.
    - Will include an ItsCompletelyHidden() constructor which will be used to store all of the hidden words from every scripture that was given to the user.

Attached below is the diagram of what my team came up with for each of our programs: A picture containing text, receipt, diagram, screenshot

Description automatically generated