W03 Assignment: Explain Encapsulation : Diogo Rangel Dos Santos

Your response must:

- Explain the meaning of Encapsulation.
- Highlight a benefit of Encapsulation.
- Provide an application of Encapsulation.
- Use a code example of Encapsulation from the program you wrote. (You should copy and paste a few lines of code that demonstrate the use of the principle.)
- Thoroughly explain these concepts. (This likely cannot be done in less than 100 words.)

Encapsulation is one of all principles of object-oriented programming (OOP). Encapsulation is a good practice of variables and the methods that operate on that data into a single unit or class while restricting direct access to some of the object's details. Using access modifiers like private, protected, and public.

One benefit of encapsulation is integrity and data security. By restricting direct access to class attributes, we can prevent unintended modifications and enforce rules on how data is accessed and updated. This leads to more maintainable and modular code, making debugging and scaling easier.

Encapsulation is commonly used in software design to create reusable and reliable components. One example is **Scripture Memorizer Program**, where we encapsulate the scripture reference and its associated text into a Scripture class. This ensures that other parts of the program cannot directly modify its data without following controlled access methods.

In this program, we encapsulated the scripture details within a Scripture class:

```
class Scripture

{

   private Reference reference;

   private string text;

   private List<string> words;

   private HashSet<int> hiddenIndices = new HashSet<int>();


   public Scripture(Reference reference, string text)

   {

      this.reference = reference;

      this.text = text;

      this.words = text.Split(' ').ToList();

   }
```

```
    public void HideRandomWords(int count)
    {
        Random rand = new Random();
        for (int i = 0; i < count; i++)
        {
            int index;
            do
            {
                index = rand.Next(words.Count);
            } while (hiddenIndices.Contains(index));

            hiddenIndices.Add(index);
            words[index] = new string('_', words[index].Length);
        }
    }

    public void Display()
    {
        Console.Clear();
        Console.WriteLine($"{reference.ToString()} - {string.Join(" ", words)}");
    }

    public bool IsFullyHidden()
    {
        return hiddenIndices.Count == words.Count;
    }
}
```