CSE 210: Programming with Classes W03 Assignment: Explain Encapsulation

What is encapsulation and why is it important?

Encapsulation is one of the core principles of object-oriented programming (OOP). It means bundling the data (variables) and the methods (functions) that operate on that data into a single unit — typically a class — and restricting direct access to some of the object’s components. In simple terms, it’s like putting the logic and the data into a box and only allowing access to the box through a controlled interface.

One major benefit of encapsulation is data protection. By making fields private and exposing only what’s necessary through public methods, we protect the inner state of an object from unintended or harmful changes. This makes the program more robust and easier to maintain because each class manages its own behavior.

For example, in the Scripture Memorizer program I worked on, we applied encapsulation to the Word class. Each word knows whether it is hidden or visible, but that state can only be changed through methods like Hide() or IsHidden(). External classes don’t directly modify that internal state.

Here’s a simple snippet from the Word class:

public class Word  
{  
 private string \_text;  
 private bool \_isHidden;  
  
 public Word(string text)  
 {  
 \_text = text;  
 \_isHidden = false;  
 }  
  
 public void Hide()  
 {  
 \_isHidden = true;  
 }  
  
 public string GetDisplayText()  
 {  
 return \_isHidden ? "\_\_\_\_\_" : \_text;  
 }  
}

In this example, the \_text and \_isHidden variables are private, so they cannot be changed directly from outside the class. Instead, the class provides methods like Hide() and GetDisplayText() to allow controlled interaction with its data. This is encapsulation in action — giving each class responsibility over its own data while hiding unnecessary details from the rest of the program.

Encapsulation helped me organize my program better, keep responsibilities clear, and prevent bugs by limiting how data is accessed and changed.