# W03 Assignment: Explain Encapsulation

Encapsulation is the principle of bundling data and methods that work on that data within one unit, typically a class, while restricting direct access to some of the object's components. In simpler terms, it means hiding the internal details of how something works, and only exposing what other parts of the program need to use. This makes programs safer, cleaner, and easier to change in the future.

One of the main benefits of encapsulation is information hiding. By keeping member variables private, we prevent other parts of the code from directly changing them, which reduces errors and makes the code more flexible if we need to update how things are stored internally. For example, if we change how a scripture is stored, only the class itself needs to change, not all the other code using it.

In our W03 Scripture Memorizer project, encapsulation is applied by making attributes like `\_text` and `\_isHidden` in the `Word` class private, and controlling access through methods such as `Hide()` or `GetDisplayText()`. This way, the `Scripture` class doesn’t worry about how words hide themselves—it just calls the public methods.

### Code Example (from Scripture Memorizer)

```csharp

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 accessed directly from outside. Instead, the class provides methods (`Hide()` and `GetDisplayText()`) to safely interact with the word. This makes the program easier to maintain and more reliable.

---

\*\*Word Count:\*\* ~200