Encapsulation

Encapsulation is one of the fundamental principles of Object-Oriented Programming. It refers to the practice of hiding the internal complexities of an object and only exposing a controlled interface for interaction. By doing so, encapsulation ensures that the internal representation of an object is protected from unintended modifications and that its behavior remains consistent and predictable.

One key benefit of encapsulation is data integrity. By restricting direct access to the object’s data, we can enforce rules and validation within the class, preventing accidental or malicious modifications. This leads to more maintainable and modular code, as changes to the internal implementation do not affect external classes that interact with it.

Encapsulation is widely used in software design patterns, particularly in abstraction and data security. For example, in a banking system, an account balance should not be directly accessible or modifiable from outside the class. Instead, it should be modified only through controlled deposit and withdrawal methods, ensuring proper validation.

In the **Scripture** class, we use private fields (\_reference and \_words) to store data, ensuring that they cannot be accessed or modified directly from outside the class. Instead, we provide controlled access through public methods, demonstrating encapsulation.

public class Scripture

{

private Reference \_reference;

private List<Word> \_words;

public Scripture(Reference reference, string text)

{

}

public void HideRandomWords(int numberToHide)

{

}

public string GetDisplayText()

{

}

public bool IsCompletelyHidden()

{

}

}