Encapsulation is one of the principles of Object Oriented Programming.

This principle establishes that we should avoid direct access to our class data such as member variables and internal methods. It is the art of keeping our internal data hidden, by defining access restrictions (private or protected data) and handling that data with class methods, getters, and setters.

Some of the benefits of encapsulations are:

- Avoid accidental modifications: By keeping our internal data private, we can determine which data can be modified and how it must be modified.
- Organization: By having methods, getters, and setters to handle the class information, we can add specific documentation (docstrings) to those accessors that serve as guides on how to treat our data.
- Validation: Methods and setters allow us to add additional validation to data modification processes.

This is an example of encapsulation:

```
public class Scripture
{
    private Reference _reference;
    private string _text;
```

The class reference has 2 private members.

If we try to access directly to the "_text" variable, an error is triggered:

```
    Scripture newScripture = new Scripture();
    newScripture._text = "This is the new text";
```

```
'Scripture._text' is inaccessible due to its protection level
[Develop03]csharp(CS0122)
```

We can define a method to access that data:

```
    public string Text
    {
        get {return _text;}
        set {_text = value;}
    }
```

And we can use that accessor to modify the data:

```
    newScripture.Text = "This is the new text";
```