Capsulation is the capability of compartmentalizing data into single units called classes. It is the principle of thinking carefully about what the public interface should be and ensuring that only necessary methods and properties are accessible from outside the class. Capsulation is important because it restrings, by hiding the details of a class, “external” access. If a program brakes it’s easier to understand where the problem is and if changes are required, there’s no need to change the code everywhere. This allows to protect the private information form the public. A good example of encapsulation are the classes I developed in my “Scripture Memorizer” program for this week. (Scripture, Word and Reference classes)

I encapsulated the implementation details of each class, hiding them from the rest of the program.

As example of code:  
class Scripture

{

private readonly Reference reference;

private readonly List<Word> words;

private readonly List<Word> originalWords;

// Constructor and other methods...

public Reference GetReference()

{

return reference;

}

public IEnumerable<Word> GetWords()

{

return words;

}

public void HideRandomWord()

{

// Method implementation...

}

public void RevealAllWords()

{

// Method implementation...

}

}