**Encapsulation**

Encapsulation is when you want to close something into something. In programming with classes, we can use the principle of encapsulation when we want to create a new class and add more than one attribute to this object. It helps better to use these attributes in another class; we call the complete constructor that contains this data. Using encapsulation, we also hide information for other classes and that can help to know if the code is broken or needs to change something, we will have to verify this class that is having problems.

We can modify an attribute/variable to be private, to just get access to it when we call that from the public constructor. The other access modifiers that want to use this private code must be in public or it won’t be able to use that part of code.

Talking about private, all attributes must be private.

One benefit of encapsulation is, we can use different workspaces as we’ve been doing, but with different attributes in each class, protecting it as it will be in private, and we can use it in other classes just calling the constructor and making this public constructor.

One way to use encapsulation is when we use the setting and getting. We can define the setting and getting of an attribute in a class, for example class Game.

In this class name will have the name of the game, style, etc…

Using a GetGameName we can obtain this name and return the name, then we can use a SetGameName and use this name to other classes.

For example, if we have the program main, and we need to show the game name, we use the constructor object, and we call the game name like that:  
  
Game newGame = new Game();

newGame.SetGameName(“God of War”);

Console.WriteLine(newGame.GetGameName());

In this code made this week in the Scripture activity, we used the principle of encapsulation:

    public string GetDisplay()

    {

        string refe = \_reference.Display() + "\n";

        for (int i = 0; i < \_wordList.Count; i++)

        {

            refe += \_wordList[i].Display() + " ";

        }

        return refe;

    }

In this part of code, we are using the class reference, that was defined as variable refe. This reference is using a Display(), the one using private variable called in a constructor. And because of encapsulation we can use this display in another class and showing this information in the main when this class Get Display() will be called.