Encapsulation:

The customers or other classes inside our program can use attributes and behaviors, but we don’t modify or manipulate them. Then, encapsulation allows the developer to protect and hide the implementation details. In this way, users or classes see and use the details needed to perform a task.

Application:

Artificial vision robot. In this process, you could have different classes used to read and save pictures. Identify colors and forms, and in one class, make decisions based on information obtained from the last class. In this way, encapsulation allows us to avoid the modification of the picture, even if the user could choose the criteria that determine the classification of the images. For example, the user wants to separate fruits by color and indicate it to the robot, but the user doesn´t need to know how the robot will do it, or the user shouldn’t change the logic to manipulate the image.

Code Example:

public class Word

{

    private string \_text = "";

    private bool \_isHidden = false;

    public Word(string text)

    {

        \_text = text;

    }

    public Word(string text, bool isHidden)

    {

        \_text = text;

        \_isHidden = isHidden;

    }

    public void ShownWord()

    {

        // Show the word if it wasn't hidden

        \_isHidden = false;

    }

    public void HiddenWord()

    {

        // Hide the word by setting \_isHidden to true

        // and replacing the text with underscores

        \_isHidden = true;

        \_text = string.Join(" ", new string('\_', \_text.Length).ToCharArray());

    }