**Encapsulation**

The principle of encapsulation is to encapsulate related data and behavior maintaining them as private as possible in a single unit through the use of classes and objects. This provides a clear interface for interacting with the encapsulated data. Splitting the program into parts and making it private can directly help the program’s organization, hiding or making the data does not mean that making the data private will encrypt the program preventing it from hackers. Instead, it will limit the visible access to some parts of our program making it more organized and having a single functionality for each class, and providing a clear structure delineating the responsibilities of each class. Also, encapsulation facilitates the code readability since each is supposed to perform one task which will make the code more organized and easy to read. Concluding, the use of encapsulation facilitates code reusability, the classes can be reused in different parts of the program leading to more efficient development and avoiding code duplication such as doing the same class, object, or function functionality two times.

For example:  
  
public class Sum

{

private int result;

public string Result

{

get { return result.ToString(); }

}

public Sum(int num1, int num2)

{

result = num1 + num2;

}

}

The example above shows the class Sum that performs one task summing a number and it has its object private, the public Result property provides an interface to access the encapsulated result, allowing external code to retrieve the result as a string without exposing the underlying implementation details. The code is clean, readable, and easy to be manipulated.