**Encapsulation**

Encapsulation is the process of bundling attributes and methods inside a class while restricting direct access to those attributes by other parts of the program different from the class. Also, encapsulation involves providing methods (within the class) that can access and manipulate those restricted attributes. These methods are commonly called access modifiers or getters and setters. This restriction is done using the **private** keyword when declaring an attribute.   
  
An attribute declared with the ‘private’ keyword cannot be access directly outside of the class. For example, consider the following example:  
  
public class Person   
{

Private int \_age;

}  
  
Person person = new Person();  
person.\_age = 20; // this would not work.  
  
The above example shows a simple Person class with an age attribute declared as private. Any attempt to assign a value to age after creating an instance of the Person class will give this sample error: ‘Person.\_age' is inaccessible due to its protection level. This demonstrates how encapsulation can restrict access to data.  
  
As earlier mentioned, encapsulation does not only involve restriction of access to attributes, it also includes carefully thinking out and providing method through which other parts of the program can access and modify attributes in the class. An example would be to provide two methods GetAge() and SetAge() which would be used return the value of the age and to assign a value to the age attribute. Still using the Person class example:

public class Person   
{

Private int \_age;

Public int GetAge()  
 {  
 return \_age;  
 }

Public void SetAge(int age)  
 {  
 \_age = age;  
 }

}  
  
Person person = new Person();  
person.SetAge(20);  
Console.WriteLine(person.GetAge()); // this would print 20  
  
  
Encapsulation provides several benefits including preventing mistakenly changing values of attributes, easing any change in the implementation of a part of the program, and streamlining the protocol of interacting with parts of our program by using methods that perform specific actions/interactions.