**C# | Encapsulation**

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In a different way, encapsulation is a protective shield that prevents the data from being accessed by the code outside this shield.

* Technically in encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of own class in which they are declared.
* As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.
* **Encapsulation can be achieved by:** Declaring all the variables in the class as private and using [C# Properties](https://www.geeksforgeeks.org/c-properties/) in the class to set and get the values of variables.

**Example:**

|  |
| --- |
| *// C# program to illustrate encapsulation*  using System;    public class DemoEncap {    *// private variables declared*  *// these can only be accessed by*  *// public methods of class*      private String studentName;      private int studentAge;    *// using accessors to get and*  *// set the value of studentName*      public String Name      {          get          {              return studentName;          }          set          {              studentName = value;          }      }    *// using accessors to get and*  *// set the value of studentAge*      public int Age      {          get          {              return studentAge;          }          set          {              studentAge = value;          }        }  }    *// Driver Class*  class GFG {  *// Main Method*      static public void Main()      {    *// creating object*          DemoEncap obj = new DemoEncap();    *// calls set accessor of the property Name,*  *// and pass "Ankita" as value of the*  *// standard field 'value'*          obj.Name = "Ankita";    *// calls set accessor of the property Age,*  *// and pass "21" as value of the*  *// standard field 'value'*          obj.Age = 21;    *// Displaying values of the variables*          Console.WriteLine("Name: " + obj.Name);          Console.WriteLine("Age: " + obj.Age);      }  } |

**Output:**

Name: Ankita

Age: 21

**Explanation:** In the above program the class *DemoEncap* is encapsulated as the variables are declared as private. To access these private variables we are using the Name and Age accessors which contains the get and set method to retrieve and set the values of private fields. Accessors are defined as public so that they can access in other class.

**Advantages of Encapsulation:**

* **Data Hiding:** The user will have no idea about the inner implementation of the class. It will not be visible to the user that how the class is stored values in the variables. He only knows that we are passing the values to accessors and variables are getting initialized to that value.
* **Increased Flexibility:** We can make the variables of the class as read-only or write-only depending on our requirement. If we wish to make the variables as read-only then we have to only use Get Accessor in the code. If we wish to make the variables as write-only then we have to only use Set Accessor.
* **Reusability:** Encapsulation also improves the re-usability and easy to change with new requirements.
* **Testing code is easy:**Encapsulated code is easy to test for unit testing.