**OPPS have four pillars –**

1. **Encapsulation**
2. **Abstraction**
3. **Inheritance**
4. **Polymorphism – overloading and overriding**

**What is Encapsulation in C#?**

The process of binding the data and functions together into a single unit (i.e. class) is called encapsulation in C#. Or you can say that the process of defining a class by hiding its internal data member direct access from outside the class and providing its access only through publicly exposed methods (setter and getter methods) or properties with proper validations and authentications is called encapsulation.

**Note:** Data encapsulation is also called data hiding because by using this principle we can hide the internal data from outside the class.

**How can we implement encapsulation in C#?**

In C# encapsulation is implemented

1. By declaring the variables as private (to restrict its direct access from outside the class)
2. By defining one pair of public setter and getter methods or properties to access private variables.

We declare variables as private to stop accessing them directly from outside the class. The public setter and getter methods or properties are used to access the private variables from outside the class with proper validations. If we provide variables access directly then we cannot validate the data before storing it in the variable.

So the point that you need to remember is by implementing encapsulation in c#, we are protecting or you can say securing the data.

**Encapsulation in C# using Accessors and Mutators:**

Let us see an example to understand this concept. In the following example, we declare the balance variable as private in the Bank class, and hence it can not be accessed directly outside of the Bank class. In order to access this balance variable, we have exposed two public methods i.e. getBalance and setBalance. The getBalance method (Accessors) to used to fetch the value store in the balance variable whereas the setBalance method (Mutator) is used to set the value in the balance variable.

##### ****What are the advantages of providing variable access via setter and getter methods in C#?****

We can validate the user-given data before it is storing in the variable. In the above program for balance variable –ve value is not allowed. So we can validate the given amount value before storing it in the balance variable. If we provide direct access to the balance variable it is not possible to validate the given amount value.

##### ****What is the problem if we don’t follow encapsulation in C# while designing a class?****

If we don’t the encapsulation principle while designing the class, then we cannot validate the user-given data according to our business requirement as well as it is very difficult to handle future changes.

Let us understand this with an example. Assume in the initial project requirement, the client did not mention that the application should not allow the negative number to store in that variable. So, we give direct access to the variable and the user can store any value to it as shown in the below program.