Encapsulation

In programming languages, encapsulation is used to refer to one of two related but distinct notions, and sometimes to the combination thereof:

- A language mechanism for restricting access to some of the object's components.
- A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.

Some programming language researchers and academics use the first meaning alone or in combination with the second as a distinguishing feature of object-oriented programming, while other programming languages which provide lexical closures view encapsulation as a feature of the language orthogonal to object orientation.

The second definition is motivated by the fact that in many OOP languages hiding of components is not automatic or can be overridden; thus, information hiding is defined as a separate notion by those who prefer the second definition.

Information hiding mechanism

Under this definition, encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. Typically, only the object's own methods can directly inspect or manipulate its fields. Languages like C# offer the programmer a degree of control over what is hidden, typically via keywords like *public* and *private*. Information hiding is accomplished by furnishing a compiled version of the source code that is interfaced via a header file.

Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. A supposed benefit of encapsulation is that it can reduce system complexity, and thus increase robustness, by allowing the developer to limit the inter-dependencies between software components.

Below is an example in C# that shows how access to a data field can be restricted through the use of a private keyword:

```
public class Account
{
        private decimal accountBalance = 500.00m;

        public decimal CheckBalance()
        {
            return accountBalance;
        }
}
static void Main()
{
        Account myAccount = new Account();
        decimal myBalance = myAccount.CheckBalance();

}
```

The process of combining elements to create a new entity. For example, a Method is a type of encapsulation because it combines a series of computer instructions. Likewise, a complex data type, such as a record or class, relies on encapsulation. Object-oriented programming languages rely heavily on encapsulation to create high-level objects. Encapsulation is closely related to abstraction and information hiding.

Why to use Encapsulation

Encapsulation means protecting important data inside the class which we do not want to be exposed outside of the class. Let's consider example of a Television(TV).

If you have seen important TV machine, TV connections and TV colour tube is hidden inside the TV case which is not been exposed for viewers like us and exposed only necessary things of a TV like TV Channel keys, TV volume keys, ON/OFF switch, Cable Switch and TV remote control for viewers to use it. This means TV machine, TV connections and TV colour tube is an unwanted data and not needed for viewers to see is been hidden from outside the world.

So encapsulation means hiding the important features of a class which is not been needed to be exposed outside of a class and exposing only necessary things of a class.

Example of Encapsulation using C#

```
class clsTelevision

        private void TVmachine()
                Console.WriteLine("Machine of a Television");

        private void TVcolortube()
                Console.WriteLine("Color Tube of a Television");

        public void TVKeys()
                Console.WriteLine("Keys of a Television");

        public void TVRemote()
                Console.WriteLine("Remote of a Television");

        public void TVScreen(){
                Console.WriteLine("Wide Screen of a Television");
```

So as you can see from our above code that we have hidden core part of a television method by using "private" access modifier and exposed only necessary methods for a viewers to use it using "public" access modifier.