CHAPTER4 : ESSENCE OF OBJECTS AND CLASSES

Objects and classes in OOP

A class is a template definition of the method s and variable s in a particular kind of object. Thus, **an object** is a specific instance of a class; it contains real values instead of variables.

Importance of objects and classes in OOP

Classes and objects provide a way to modularize programming code and encapsulating functionality.
One of my favorite dimensions to OOP is the habit of "Black Boxing" programming functionality. By making methods and properties private, programmers are able to bundle programming code up into a module that other programmers don't have to know anything about. They simply have to know what services the object offers and how to access them.

Another important dimension of OOP is "loose coupling." Well-written classes make it so that it is possible to easily replace portions of programming code without impacting the rest of the system. In a well-written OOP architecture, you can switch databases without having to touch the code in your control or view layers.

Implementation of objects and classes

✓ **Initialization**

Initialization is the assignment of an initial value for a data **object** or **variable**. The manner in which **initialization** is performed depends on programming language, as well as type, storage class, etc., of an **object** to be **initialized**.

The variable's *name* is what you ***declare*** it to be. The *value* is what you ***assign*** to it.

Variables are initialized

All variables are always given an *initial* value at the point the variable is *declared*. Thus all variables are ***initialized***.

For *value* types, like `int` the compiler will give them a valid value if you do not do so explicitly. `int`'s ***initialize*** to zero by default, `DateTime`'s ***initialize*** to `DateTime.MinValue` by default.

Reference type variables ***initialize*** to the object you give it. The compiler will not ***assign*** an object (i.e. a valid value) if you don't. In this case the value is `null` - nothing. So we say that the reference is ***initialized*** to null.