C# Encapsulation Documentation

Encapsulation

- Encapsulation is the process of encapsulating data and functions into a single unit (for e.g. a class).
- The need for encapsulation is to protect or prevent the code (data) from accidental corruption.
- This protection can be implemented through the use of access modifiers.
- The available access modifiers in C# are applied to types or members e.g. public, private, internal, protected, protected internal, private protected.

C# Access Modifiers

- **Public:** If a type or member is public it means the type or member can be accessed by any other code in the same assembly as where the type or member is declared or any code in another assembly that references it.
- **Private:** If a type or member is private the type or member can be accessed by code resident in the same class but the member or type cannot be accessed by code external to the class in which the member or type is declared.
- **Internal:** If a type or member is internal the type or member can be accessed by any code within the same assembly but cannot be accessed by code within another assembly.
- **Protected:** If a type or member is protected the type or member can only be accessed by code within the class in which the member or type is declared or a class derived from the class in which the type or member is declared.
- **Protected Internal:** if a type or member is protected internal the type or member can be accessed by any code in the assembly in which it is declared, or from within a derived class in another assembly.
- **Private protected:** If a type or member is private protected the type or member can be accessed only from within the assembly in which the type or member is declared, by code in the same class or from within a type that is derived from the class in which the private protected member or type is declared.

Encapsulation is concerned with the protection against unintended access to data belonging to a class member or type.

Abstraction is a design principle used for abstracting an interface for an object, in order to decouple design implementation from code logic implementation detail.

Inheritance allows for a class to be based on another class.

Polymorphism is the provision of a single interface to entities of different types.

The word, "polymorphism", has Greek roots.
"poly", meaning "many"
"morph" - meaning "form"


gl
Digital