OOP Concept – Encapsulation

Overview

Encapsulation allows an object to separate its interface from its implementation. The data and the implementation code for the object are hidden behind its interface.

Encapsulation hides internal implementation details from users.

Example:

A Customer may issue a check and now know how it is processed. The internal processing is hidden from the customer. Similary, the inessential attributes of a class are hidden from users by using encapsulation. The hidden attributes of a class are called protected attributes.

It ensures that an object supplies only the requested information to another object and hides inessential information.

Example: *When a user selects a command from a menu in an application, the code used to perform the actions of that command is hidden from the user.*

Encapsulation packages the data and method of an object and provides protection from external tampering by users. It implies that there is visibility to the functionalities offered by an object, and no visibility to its data. The best application of encapsulation is making the data fields private and using public accessor methods.

However, you cannot hide an entire object. To use an object, a part of it needs to be accessed by users. To provide this access, abstraction is used. Abstraction provides access to a specific part of

data while encapsulation hides the data. Therefore, abstraction and encapsulation compliment each other.

In OOP, abstraction defines the conceptual boundaries of an object. These boundaries distinguish an object from another objects.

Example: *When a user designs an application by using existing templates, the complexity of the template is hidden from the user, but the essential features for creating the application are provided to the user. Abstraction enables an object to display these essential features to develop an application.*

Encapsulation and Inheritance are two key concepts in OOP that serve different purposes. While encapsulation hides inessential information, inheritance allows the object of a class to adopt the attributes of another class.

When you implement inheritance in an application, the classes in the application are arranged in a strict hierarchy. The classes at the lower levels inherit the attributes of the classes at the higher levels.

In addition to inheriting the characteristics of the parent class, a class may have its own attributes. This feature implies that the code and characteristics of a class are reusable and extensible. However, inheritance compromises encapsulation because a subclass can directly access the parent's protected attributes without using operations.

Source : http://idynsolutions.wordpress.com/2006/11/08/oop-concept-encapsulation/