Discussion 10

How Inheritance weakens / breaks Encapsulation

<https://www.informit.com/articles/article.aspx?p=2210836>

Inheritance weakens encapsulation because the two concepts contradict each other. Encapsulation focuses on hiding information, while inheritance requires at least some of the class’s information to be made known to its child classes. Changing the implementation of a parent class directly impacts the corresponding subclasses, in what was described as “a ripple effect” in the article. Therefore, these implementation changes of the base class are not necessarily encapsulated in the derived classes. If you ensure to use these two concepts in a well thought out and appropriate manner, then a lot of the dilemmas that this topic uncovers could potentially be avoided. In other words, it’s important to use inheritance appropriately and responsibly. Ensuring each derived class should actually be an extension of the base class can prevent the code from breaking if/when you implement changes to that base class.

Notes :

Stephen Gilbert and Bill McCarty define encapsulation as “the process of packaging your program, dividing each of its classes into two distinct parts: the interface and the implementation.”

Encapsulation is the process of packaging classes into the public interface and the private implementation.

Peter Coad and Mark Mayfield make a case that when using inheritance, encapsulation is inherently weakened within a class hierarchy. They talk about a specific risk: Inheritance connotes strong encapsulation with other classes but weak encapsulation between a superclass and its subclasses.

As stated in Chapter 1, polymorphism literally means many shapes. When a message is sent to an object, the object must have a method defined to respond to that message. In an inheritance hierarchy, all subclasses inherit the interfaces from their superclass. However, because each subclass is a separate entity, each might require a separate response to the same message.