**DETAILED DESCRIPTION OF DESIGN PATTERNS USED FOR NOTIFICATION**

Adapter Pattern (INotifierAdapter)

An adapter helps two incompatible interfaces to work together. This is the real world definition for an adapter. Interfaces may be incompatible but the inner functionality should suit the need. The Adapter design pattern allows otherwise incompatible classes to work together by converting the interface of one class into an interface expected by the clients.

Factory Pattern (NotificationFactory)

The factory method pattern is a [creational pattern](http://en.wikipedia.org/wiki/Creational_pattern) which uses factory methods to deal with the problem of [creating objects](http://en.wikipedia.org/wiki/Object_creation) without specifying the exact [class](http://en.wikipedia.org/wiki/Class_(computer_science)) of object that will be created. This is done by creating objects via calling a factory method—either specified in an interface and implemented by child classes, or implemented in a base class and optionally overridden by derived classes

Singleton Pattern (NotificationFactory)

 The singleton pattern is a [design pattern](http://en.wikipedia.org/wiki/Design_pattern_(computer_science)) that restricts the [instantiation](http://en.wikipedia.org/wiki/Instantiation_(computer_science)) of a class to one [object](http://en.wikipedia.org/wiki/Object-oriented_programming). This is useful when exactly one object is needed to coordinate actions across the system. The concept is sometimes generalized to systems that operate more efficiently when only one object exists, or that restrict the instantiation to a certain number of objects.

