Observer Design Pattern:

->The purpose of observer pattern is to notify the interested observers about some change that occurred

->we can also add more observers during runtime as well as remove them

Example:

->We have a form to select the colour.For colour change we have to update the entire application.So there will be observers listening to the colour change event for updating themselves.

Subject and Observers:

->The two important key terms in this pattern are subject and observer

->Subject:

->Subject is the object which holds the value and takes responsibility in notifying the observers when the value is changed.The subject could be the database change , property change , key change of text box etc. and so on

->So there will be two interfaces

->Subject Interface:

public Interface ISubject

{

void Register();

void UnRegister();

void Notify();

}

->Observer:

->The observer is the object listening to the subject's change.

->It will have its own updation routine which It will run when it is notified by the subject

->Observer Interface:

public interface IObserver

{

void ColorChanged(Color newColor);

}

->So in the above example , we are using an observer interface which has a colour changed method.So the interested observers should implement this interface to get notified.

->There will be only one Subject and multiple number of observers

->Registering and Unregistering:

->In the above interface,the observer can use the Register() method to get notified about the changes

->At anytime it can unregister about notifications using the Unregister() method

->Notify():

->The notify method will take care of calling the listening observers.

Subject

|

Observer1 Observer2 … Observer N

->Implementation using code:

->public class ColorSubject : ISubject

{

private Color \_Color=Color.Blue;

public Color Color

{

Get

{ return \_Color; }

Set

{

\_Color=value;

Notify();

}

}

    #region ISubject Members

private HashSet<IObserver> \_observers = new HashSet<IObserver>();

public void Register(IObserver observer)

{

\_observers.Add(observer);

}

Public void UnRegister(IObserver observer)

{

\_observers.Remove(observer);

}

Public void Notify()

{

\_observers.ToList().ForEach(o=>o.ColorChanged(Color));

}

}