# Design Pattern

## Observer Pattern:

|  |
| --- |
| namespace ObserverPattern  {    public interface IInvestor    {      void UpdatePrice(IStock stock);    }  } |
| using System;    namespace ObserverPattern  {    public class Investor : IInvestor    {      private string \_name;        #region IInvestor Members        public Investor(string name)      {        Name = name;      }        public string Name      {        get { return \_name; }        set { \_name = value; }      }        public void UpdatePrice(IStock stock)      {        Console.WriteLine(string.Format("Investor: {0},\tStock: {1},\tPrice: {2}", Name, stock.Symbol, stock.Price));      }        #endregion    }  } |
| namespace ObserverPattern  {    public interface IStock    {      double Price { get; set; }      string Symbol { get; }          void RegisterInvestor(IInvestor investor);      void RemoveInvestor(IInvestor investor);      void NotifyInvestor();    }  } |
| using System.Collections.Generic;    namespace ObserverPattern  {    public class Stock : IStock    {      IList<IInvestor> \_investors = new List<IInvestor>();      private string \_symbol = string.Empty;      private double \_price = 0;        public Stock(string symbol, double price)      {        Price = price;        \_symbol = symbol;      }        #region IStock Members        public string Symbol      {        get { return \_symbol; }      }        public double Price      {        get { return \_price; }        set        {          \_price = value;          this.NotifyInvestor();        }      }        public void RegisterInvestor(IInvestor investor)      {        \_investors.Add(investor);      }        public void RemoveInvestor(IInvestor investor)      {        \_investors.Remove(investor);      }        public void NotifyInvestor()      {        foreach (var investor in \_investors)        {          investor.UpdatePrice(this);        }      }        #endregion    }  } |
| using System;  using System.Collections.Generic;  using System.Linq;    namespace ObserverPattern  {    class Program    {      static void Main(string[] args)      {        var stockList = new List<IStock>();        IStock ibmStock = new Stock("IBM", 12.34);        IStock hclStock = new Stock("HCL", 02.98);        IStock pslStock = new Stock("PSL", 16.67);        IStock msStock = new Stock("MS", 20.3);          stockList.Add(ibmStock);        stockList.Add(hclStock);        stockList.Add(pslStock);        stockList.Add(msStock);          IInvestor investor01 = new Investor("Siddharth");        ibmStock.RegisterInvestor(investor01);        msStock.RegisterInvestor(investor01);          IInvestor investor02 = new Investor("Pradeep");        pslStock.RegisterInvestor(investor02);          IInvestor investor03 = new Investor("Richa");        msStock.RegisterInvestor(investor03);        pslStock.RegisterInvestor(investor03);          IInvestor investor04 = new Investor("Krishna Chandra Mishra");        hclStock.RegisterInvestor(investor04);        msStock.RegisterInvestor(investor04);        ibmStock.RegisterInvestor(investor04);          var exit = false;          while (!exit)        {          Console.WriteLine("Enter Stock Symbol:");          var symbol = Console.ReadLine();          Console.WriteLine("Enter Stock Price: ");          var price = Console.ReadLine();            var stock = stockList.Where(s => s.Symbol.ToLowerInvariant().Equals(symbol.ToLowerInvariant())).FirstOrDefault();          if (stock == null)            Console.WriteLine("Stock not found");          else          {            stock.Price = Convert.ToDouble(price);          }          }          Console.ReadKey();        }    }  } |