using System;

using System.Collections.Generic;

using System.Threading;

namespace RefactoringGuru.DesignPatterns.Observer.Conceptual

{

public interface IObserver

{

void Update(ISubject subject);

}

public interface ISubject

{

void Attach(IObserver observer);

void Detach(IObserver observer);

void Notify();

}

public class Subject : ISubject

{

public int State { get; set; } = 0;

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

public void Attach(IObserver observer)

{

Console.WriteLine("Subject: Attached an observer.");

this.\_observers.Add(observer);

}

public void Detach(IObserver observer)

{

this.\_observers.Remove(observer);

Console.WriteLine("Subject: Detached an observer.");

}

public void Notify()

{

Console.WriteLine("Subject: Notifying observers...");

foreach (var observer in \_observers)

{

observer.Update(this);

}

}

public void SomeBusinessLogic()

{

Console.WriteLine("\nSubject: I'm doing something important.");

this.State = new Random().Next(0, 10);

Thread.Sleep(15);

Console.WriteLine("Subject: My state has just changed to: " + this.State);

this.Notify();

}

}

class ConcreteObserverA : IObserver

{

public void Update(ISubject subject)

{

Console.WriteLine("You got messages by SMS");

}

}

class ConcreteObserverB : IObserver

{

public void Update(ISubject subject)

{

Console.Beep(700, 1500);

Console.Beep(900, 500);

Console.WriteLine("You got messages by AirDrop");

}

}

class Program

{

static void Main(string[] args)

{

var subject = new Subject();

var observerA = new ConcreteObserverA();

subject.Attach(observerA);

var observerB = new ConcreteObserverB();

subject.Attach(observerB);

subject.Attach(observerB);

subject.Attach(observerB);

subject.SomeBusinessLogic();

subject.SomeBusinessLogic();

subject.Detach(observerB);

subject.SomeBusinessLogic();

}

}

}