TUGAS PENDAHULUAN
KONSTRUKSI PERANGKAT LUNAK
MODUL XIII
DESIGN_PATTERN_IMPLEMENTATION

Disusun Oleh:

Lintang Suminar Tyas Wening

2211104009

SE0601

Dosen Pengampu:

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

Source Code

```
using System;
using System.Collections.Generic;
using System.Threading;

namespace ObserverPatternExample
{
    // Interface Observer
    public interface IObserver
    {
        void Update(ISubject subject);
    }

    // Interface Subject (Publisher)
    public interface ISubject
    {
        void Attach(IObserver observer);
        void Detach(IObserver observer);
        void Notify();
    }

    // Concrete Subject
    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.");
            _observers.Add(observer);
        }

        public void Detach(IObserver observer)
        {
            _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();
        }
    }

    // Concrete Observer A
    public class ConcreteObserverA : IObserver
    {
        public void Update(ISubject subject)
        {
            if ((subject as Subject).State < 3)
            {
                Console.WriteLine("ConcreteObserverA: Reacted to the event.");
            }
        }
    }

    // Concrete Observer B
    public class ConcreteObserverB : IObserver
    {
        public void Update(ISubject subject)
        {
            if ((subject as Subject).State == 0 || (subject as Subject).State >= 2)
            {
                Console.WriteLine("ConcreteObserverB: Reacted to the event.");
            }
        }
    }

    // Main Program
    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.SomeBusinessLogic();  // Observer A dan B merespon
            subject.SomeBusinessLogic();  // Observer A dan B merespon

            subject.Detach(observerB);    // Observer B tidak merespon

            subject.SomeBusinessLogic();  // Observer A yang merespon
        }
    }
}
```

Hasil Output

```
Subject: Attached an observer.
Subject: Attached an observer.
Subject: I'm doing something important.
Subject: My state has just changed to: 4
Subject: Notifying observers...
ConcreteObserverB: Reacted to the event.
Subject: I'm doing something important.
Subject: My state has just changed to: 0
Subject: Notifying observers...
ConcreteObserverA: Reacted to the event.
ConcreteObserverB: Reacted to the event.
Subject: Detached an observer.
Subject: I'm doing something important.
Subject: My state has just changed to: 2
Subject: Notifying observers...
ConcreteObserverA: Reacted to the event.
```

Penjelasan

Kodenya merupakan implementasi dari *Observer Pattern* dalam bahasa C#. Pola ini digunakan untuk menciptakan hubungan satu-ke-banyak antara objek, di mana ketika objek utama (Subject) berubah, semua objek yang bergantung padanya (Observer) akan diberi tahu dan diperbarui secara otomatis. Dalam program ini, terdapat dua antarmuka utama, yaitu ISubject dan IObserver. Antarmuka ISubject berfungsi sebagai publisher yang memungkinkan objek observer untuk didaftarkan (Attach), dilepas (Detach), dan diberi notifikasi (Notify). Sementara itu, IObserver hanya memiliki satu metode, yaitu Update, yang akan dipanggil ketika terjadi perubahan pada Subject. Kelas Subject mengimplementasikan ISubject dan memiliki properti State yang nilainya diubah secara acak dalam metode SomeBusinessLogic(). Setiap kali state berubah, metode Notify() dipanggil untuk memberi tahu semua observer yang telah terdaftar. Kelas ConcreteObserverA dan ConcreteObserverB adalah dua implementasi dari IObserver yang akan bereaksi terhadap perubahan state dengan logika tertentu. Observer A hanya merespons jika state bernilai kurang dari 3, sedangkan Observer B merespons jika state bernilai 0 atau lebih besar dari atau sama dengan 2. Pada method Main, objek subject dibuat dan kedua observer ditambahkan. Kemudian, SomeBusinessLogic() dijalankan beberapa kali untuk memicu perubahan dan reaksi observer. Setelah Observer B dilepas, hanya Observer A yang akan menerima pemberitahuan perubahan selanjutnya.