TUGAS PENDAHULUAN
KONSTRUKSI PERANGKAT LUNAK
MODUL 13

Disusun oleh :
Althafia Defiyandrea Laskanadya Wibowo

S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

MENJELASKAN SALAH SATU DESIGN PATTERN

a. Berikan salah satu contoh kondisi dimana design pattern “Observer” dapat digunakan

Observer pattern sangat cocok digunakan saat terdapat satu objek yang berfungsi sebagai pusat data (subject), dan banyak objek lain yang harus mengikuti perubahan data dari objek tersebut (observers) secara otomatis. Contoh nyata: Sistem notifikasi media sosial. Ketika seorang pengguna membuat posting baru, maka semua pengikut (followers) akan secara otomatis mendapatkan pemberitahuan (notifikasi) bahwa ada posting baru.

b. Langkah-langkah Implementasi Observer Pattern

1. Buat interface Observer (IObserver)
    - Interface ini berisi method Update() yang akan dipanggil saat terjadi perubahan.
2. Buat interface Subject (ISubject)
    - Interface ini memiliki method Attach(observer), Detach(observer), dan Notify().
3. Buat class konkret Subject (ConcreteSubject)
    - Mengimplementasikan ISubject.
    - Menyimpan daftar observer dan memberi tahu mereka jika ada perubahan.
4. Buat class konkret Observer (ConcreteObserver)
    - Mengimplementasikan IObserver.
    - Menanggapi perubahan dari subject melalui method Update().
5. Di method Main() atau Run(),
    - Buat objek subject dan observer.
    - Lakukan pendaftaran (attach) observer ke subject.
    - Saat data subject berubah, panggil Notify() untuk memberi tahu semua observer.

c. Kelebihan dan Kekurangan Observer Pattern

1. Kelebihan :
    - Loose Coupling – Subject tidak perlu tahu detail dari observers.
    - Mudah ditambah observer baru tanpa ubah kode subject.
    - Cocok untuk aplikasi event-driven atau real-time.
    - Mendukung prinsip Open/Closed
2. Kekurangan :
    - Sulit di-debug – Banyak objek saling berinteraksi secara tidak langsung.

- Risiko over-notification jika terlalu banyak observer.
- Jika lupa detach observer → bisa terjadi memory leak.
- Tidak cocok jika jumlah observer sangat besar tanpa pengelolaan yang efisien.

IMPLEMENTAS KODE

```
public interface IObserver
{
    void Update(string message);
}

public interface ISubject
{
    void Attach(IObserver observer);
    void Detach(IObserver observer);
    void Notify(string message);
}

public class NewsPublisher : ISubject
{
    private List<IObserver> observers = new List<IObserver>();

    public void Attach(IObserver observer)
    {
        observers.Add(observer);
    }

    public void Detach(IObserver observer)
    {
        observers.Remove(observer);
    }

    public void Notify(string message)
    {
        foreach (var observer in observers)
        {
            observer.Update(message);
        }
    }
}
```

```
public class SubscriberA : IObserver
{
    public void Update(string message)
    {
        Console.WriteLine($"Subscriber A received: {message}");
    }
}

public class SubscriberB : IObserver
{
    public void Update(string message)
    {
        Console.WriteLine($"Subscriber B received: {message}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        var publisher = new NewsPublisher();
        var sub1 = new SubscriberA();
        var sub2 = new SubscriberB();

        publisher.Attach(sub1);
        publisher.Attach(sub2);

        publisher.Notify("Berita hari ini: Cuaca cerah!");

        publisher.Detach(sub1);
        publisher.Notify("Berita kedua: Hujan deras!");
    }
}
```

Program diatas adalah implementasi dari design pattern Observer dalam bahasa C#. IObserver adalah antarmuka yang mewakili objek-objek yang akan menerima notifikasi, sedangkan ISubject adalah antarmuka untuk objek yang mengelola dan memberi tahu observer. NewsPublisher adalah class konkret yang bertindak sebagai subject, menyimpan daftar observer, dan memanggil method Update() milik observer saat terjadi notifikasi. SubscriberA dan SubscriberB adalah observer yang masing-masing mencetak pesan ketika menerima notifikasi. Pada method Main(), dua observer didaftarkan ke NewsPublisher, lalu menerima pesan saat Notify() dipanggil. Setelah SubscriberA dihapus dari daftar, hanya SubscriberB yang menerima notifikasi selanjutnya.