# TUGAS PENDAHULUAN
# KONSTRUKSI PERANGKAT LUNAK
# MODUL XIII
# DESIGN PATTERN IMPLEMENTATION


**Disusun Oleh:**

**Maria Nathasya Desfera Pangestu**

**2211104008**

**SE0601**

**Dosen Pengampu:**

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

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**


**2025**

Tugas Pendahuluan

1. MEMBUAT PROJECT GUI BARU Buka IDE misalnya dengan Visual Studio
   A. Misalnya menggunakan Visual Studio, buatlah project baru dengan nama tpmodul113_NIM
   B. Project yang dibuat bisa berupa console atau sejenisnya
2. MENJELASKAN SALAH SATU DESIGN PATTERN Buka halaman web https://refactoring.guru/design-patterns/catalog kemudian baca design pattern dengan nama "Observer", dan jawab pertanyaan berikut ini (dalam Bahasa Indonesia):
   A. Berikan salah satu contoh kondisi dimana design pattern "Observer" dapat digunakan?
      Jawab:
      Saat ada pelanggan mau menerima pemberitahuan dari toko tentang ketersediaan produk seperti apa saja laptop terbaru yang rilis. Pelanggan tidak perlu datang ke toko untuk mengetahuinya karena toko bisa mengirimkan notifikasi kepada semua pelanggan. Namun Observer Pattern memungkinkan toko untuk mengirimkan pemberitahuan hanya kepada pelanggan yang telah berlangganan untuk menerima notifikasi tersebut.

   B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern "Observer" .
      Jawab:
      Langkah-langkah untuk menerapkan Observer Pattern dalam studi kasus berlangganan di toko laptop yaitu:
      - Tentukan objek yang berfungsi sebagai subjek(publisher) dan pengamat (subscriber).
      - Buat antarmuka untuk subscriber, setidaknya memiliki metode update() yang akan dipanggil ketika terjadi perubahan.
      - Buat antarmuka untuk publisher yang menyediakan metode attach(), detach(), dan notify() untuk mengelola daftar subscriber.
      - Implementasikan daftar langganan, di mana publisher akan menyimpan daftar subscriber dan memanggil metode update() mereka saat terjadi suatu peristiwa.
      - Kembangkan subscriber konkret, yang akan merespons notifikasi dengan mengambil data dari publisher (jika diperlukan).
      - Di sisi klien, buat dan sambungkan objek publisher dengan subscriber secara dinamis sesuai dengan kebutuhan.

   C. Berikan kelebihan dan kekurangan dari design pattern "Observer"
      Jawab:
      Kelebihan:
      - Loose Coupling
        Publisher dan Subscriber tidak memiliki ketergantungan langsung satu sama lain. Keduanya hanya terhubung melalui antarmuka, yang mempermudah proses pengembangan dan pemeliharaan.
      - Mendukung Prinsip Open/Closed
        Memungkinkan penambahan tipe Subscriber baru tanpa perlu mengubah kode Publisher, sehingga memberikan fleksibilitas lebih terhadap perubahan.

- Fleksibilitas Dinamis pada Waktu Runtime
  Objek dapat melakukan subscribe atau pembatalan unsubscribe kapan saja selama program berjalan, menjadikan pola ini sangat adaptif.
- Komunikasi yang Efisien untuk Banyak Objek
  Pola ini sangat cocok digunakan ketika satu objek perlu memberi tahu banyak objek lain mengenai perubahan status atau peristiwa tertentu.

Kekurangan:

- Kesulitan dalam Melacak Aliran Notifikasi
  Karena hubungan antara Publisher dan Subscriber bersifat tidak langsung, menjadi sulit untuk melacak urutan atau penyebab notifikasi dalam sistem yang besar.
- Potensi Masalah Kinerja
  Jika Publisher memiliki banyak Subscriber, proses pengiriman notifikasi dapat menjadi lambat, terutama jika setiap Subscriber melakukan operasi yang berat.
- Ketergantungan Tidak Langsung dari Subscriber pada Publisher
  Meskipun kelas-kelas tersebut tidak terikat langsung, implementasi notifikasi dapat menciptakan ketergantungan tersembunyi melalui data atau metode tertentu.
- Masalah Urutan Eksekusi
  Publisher tidak menjamin urutan pengiriman notifikasi kepada setiap Subscriber, yang dapat menjadi masalah jika urutan eksekusi sangat penting

3. IMPLEMENTASI DAN PEMAHAMAN DESIGN PATTERN OBSERVER Buka halaman web berikut https://refactoring.guru/design-patterns/observer dan scroll ke bagian “Code Examples”, pilih kode yang akan dilihat misalnya C# dan ikuti langkah-langkah berikut:

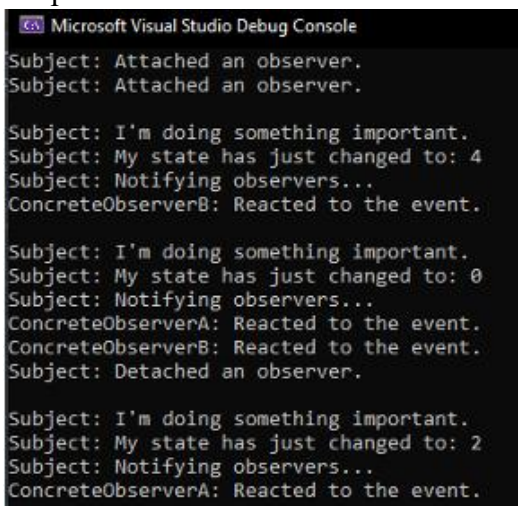

A. Pada project yang telah dibuat sebelumnya, tambahkan kode yang mirip atau sama dengan contoh kode yang diberikan di halaman web tersebut
B. Jalankan program tersebut dan pastikan tidak ada error pada saat project dijalankan
C. Jelaskan tiap baris kode yang terdapat di bagian method utama atau “main”
Jawab:
Source code

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

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

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

    // Concrete Subject
    4 references
    public class Subject : ISubject
    {
        5 references
        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)
```

```
44              {
45                  observer.Update(this);
46              }
47          }
48
            3 references
49          public void SomeBusinessLogic()
50          {
51              Console.WriteLine("\nSubject: I'm doing something important.");
52              this.State = new Random().Next(0, 10);
53
54              Thread.Sleep(15);
55
56              Console.WriteLine($"Subject: My state has just changed to: {this.State}");
57              this.Notify();
58          }
59      }
60
61      // Concrete Observer A
        1 reference
62      public class ConcreteObserverA : IObserver
63      {
            1 reference
64          public void Update(ISubject subject)
65          {
66              if ((subject as Subject).State < 3)
67              {
68                  Console.WriteLine("ConcreteObserverA: Reacted to the event.");
69              }
70          }
71      }
72
73      // Concrete Observer B
        1 reference
74      public class ConcreteObserverB : IObserver
75      {
            1 reference
76          public void Update(ISubject subject)
77          {
78              if ((subject as Subject).State == 0 || (subject as Subject).State >= 2)
79              {
80                  Console.WriteLine("ConcreteObserverB: Reacted to the event.");
81              }
82          }
83      }
84
85      // Main Program
        0 references
86      class Program
87      {
            0 references
```

```
88          static void Main(string[] args)
89          {
90              var subject = new Subject();
91
92              var observerA = new ConcreteObserverA();
93              subject.Attach(observerA);
94
95              var observerB = new ConcreteObserverB();
96              subject.Attach(observerB);
97
98              subject.SomeBusinessLogic();  // Observer A dan B merespon
99              subject.SomeBusinessLogic();  // Observer A dan B merespon
100
101             subject.Detach(observerB);    // Observer B tidak lagi merespon
102
103             subject.SomeBusinessLogic();  // Hanya Observer A yang merespon
104         }
105     }
106 }
```

Output




Penjelasan

- Membuat Subject
  subject dibuat sebagai pusat pola Observer yang menyimpan state dan memberi tahu observer saat terjadi perubahan.
- Menambahkan ObserverA
  observerA dibuat dan didaftarkan ke subject agar menerima notifikasi saat subject berubah.
- Menambahkan ObserverB
  observerB juga dibuat dan ditambahkan ke subject, sehingga akan ikut menerima notifikasi perubahan.
- Menjalankan Logika Bisnis
  Saat SomeBusinessLogic() dijalankan, subject memproses logika bisnis yang bisa mengubah state, lalu memberi tahu semua observer yang terdaftar.
- Menghapus ObserverB
  observerB dihapus dari daftar observer menggunakan Detach(), sehingga tidak lagi menerima pemberitahuan perubahan.