

TP MODUL 13



Disusun Oleh :

Nama : Ganes Gemi Putra

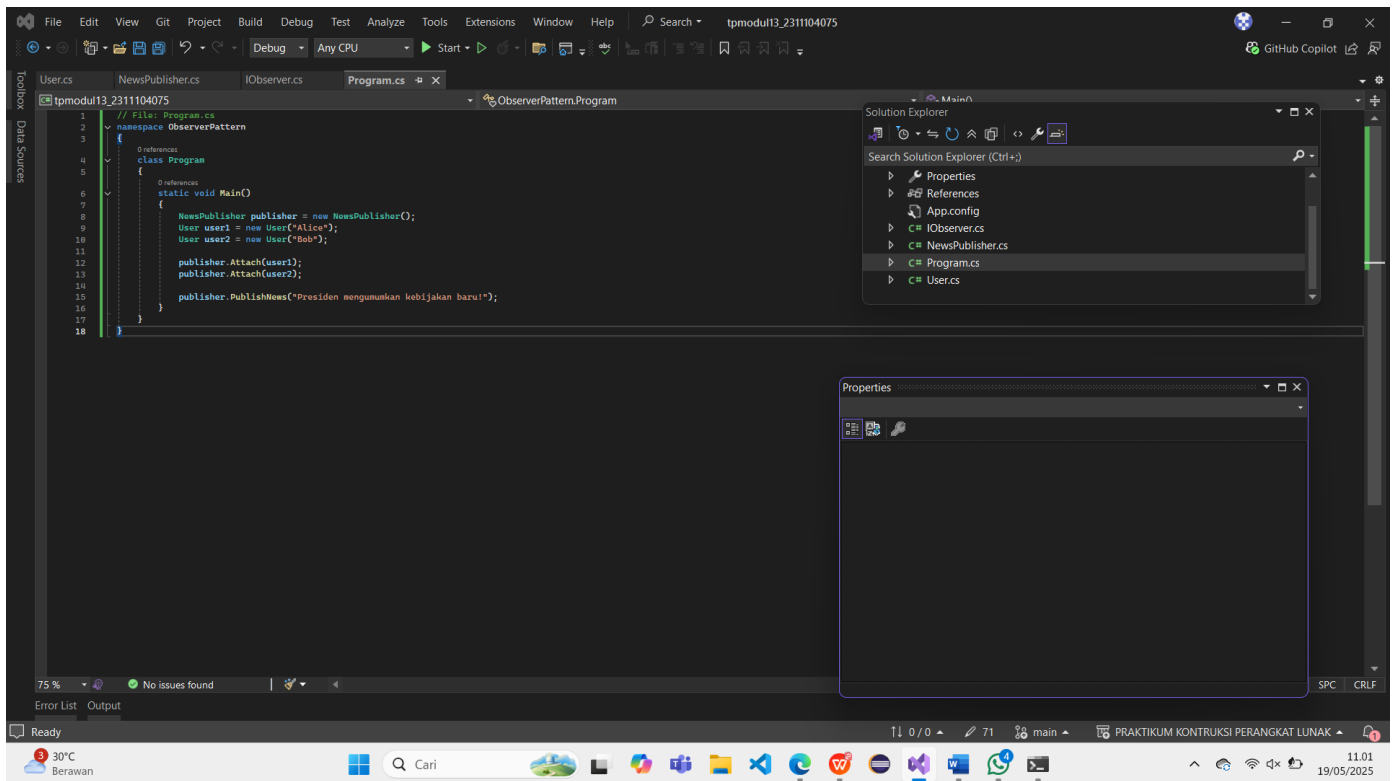
Kelas : SE-07-02

NIM : (2311104075)

Dosen : YUDHA ISLAMI SULISTYA

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2025**

1. MEMBUAT PROJECT GUI BARU



2. MENJELASKAN SALAH SATU DESIGN PATTERN (OBSERVER)

Contoh Kasus:

Sistem Notifikasi pada Aplikasi Berita.

Penjelasan:

➤ **Subject:** Aplikasi berita (misal: "[Alice] Berita terbaru: Presiden mengumumkan kebijakan baru!").

[Bob] Berita terbaru: Presiden mengumumkan kebijakan baru!

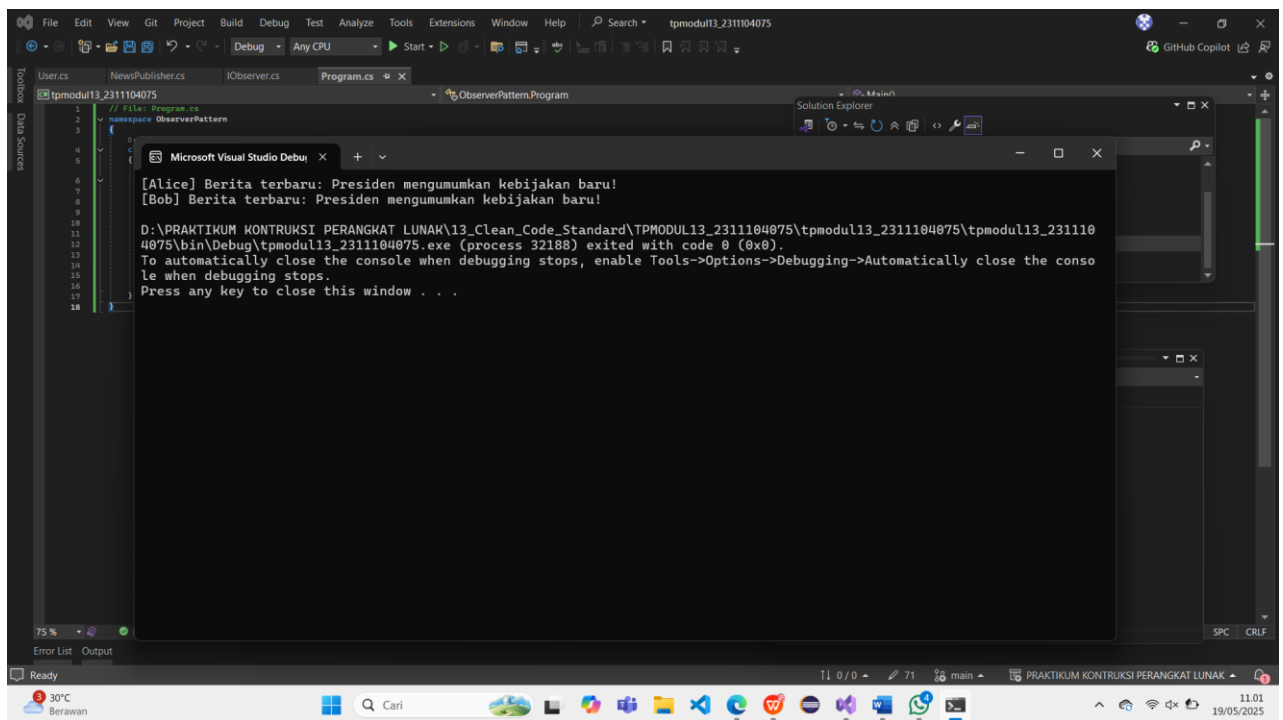
- **Observer:** Pengguna yang berlangganan notifikasi.

- **Mekanisme:**

- Ketika ada berita baru, aplikasi mengirimkan notifikasi ke semua pengguna yang berlangganan.
- Setiap pengguna (observer) menerima pembaruan secara otomatis tanpa perlu memeriksa aplikasi secara manual.

B. Langkah Implementasi Observer Pattern

Buat Interface IObserver



Interface IObserver

File: IObserver.cs

Penjelasan:

- **Tujuan:** Mendefinisikan kontrak (method) yang harus diimplementasikan oleh semua *observer*.

- **Method Utama:**

```
void Update(string message);
```

- ✓ Dipanggil oleh *subject* (NewsPublisher) untuk mengirim pembaruan ke *observer*.
- ✓ Parameter message: Data/notifikasi yang dikirim ke observer.

Contoh Analogi:

- Seperti "kontrak langganan" yang harus disetujui oleh pelanggan (*observer*) untuk menerima berita.

🚩 Class NewsPublisher (Subject/Publisher)

File: NewsPublisher.cs

Penjelasan:

- ❖ **Tujuan:** Bertindak sebagai *subject* yang mengelola daftar observer dan mengirim notifikasi saat ada perubahan.

- **Komponen Utama:**

- **Daftar Observer:**

```
private List<IObserver> _subscribers = new List<IObserver>();
```

Menyimpan semua observer yang berlangganan

Method Attach() dan Detach():

```
public void Attach(IObserver subscriber) => _subscribers.Add(subscriber);
```

```
public void Detach(IObserver subscriber) => _subscribers.Remove(subscriber);
```

- ✓ Menambah/menghapus observer dari daftar.

Method PublishNews():

```
public void PublishNews(string news)
{
    _latestNews = news;
    Notify();
}
```

- ✓ Memicu notifikasi ke semua observer saat ada berita baru.

Contoh Analogi:

- Seperti "stasiun berita" yang mengirim update ke semua pelanggan ketika ada berita terbaru.

Class User (Concrete Observer)

File: User.cs

Penjelasan:

- **Tujuan:** Implementasi konkret dari observer yang menerima notifikasi.
- **Method Utama:**

Constructor:

```
public User(string name) => _name = name;
```

- Inisialisasi nama pengguna.

Method Update():

```
public void Update(string news)
{
    Console.WriteLine($"[{_name}] Berita terbaru: {news}");
}
```

- Logika yang dijalankan saat menerima notifikasi dari *subject*.

Contoh Analogi:

- Seperti "pelanggan aplikasi berita" yang menerima notifikasi di ponsel.

Class Program (Main Program)

File: Program.cs

Penjelasan:

- **Tujuan:** Menghubungkan semua komponen dan menjalankan program.
- **Alur Kerja:**
 1. Membuat objek NewsPublisher (subject).
 2. Membuat objek User (observer).
 3. Menghubungkan observer ke subject dengan Attach().
 4. Memicu notifikasi dengan PublishNews().

Contoh Kode:

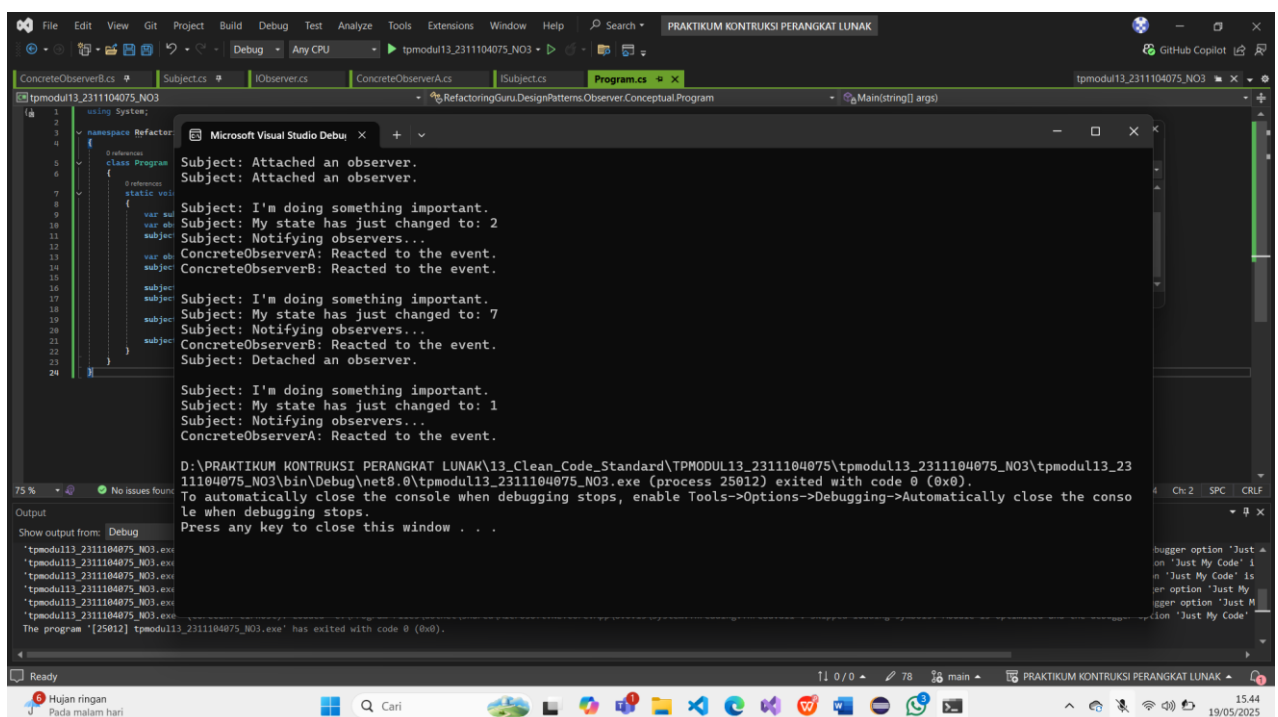
```
NewsPublisher publisher = new NewsPublisher();  
User user1 = new User("Alice");  
User user2 = new User("Bob");  
  
publisher.Attach(user1);  
publisher.Attach(user2);  
  
publisher.PublishNews("Presiden mengumumkan kebijakan baru!");
```

Diagram Interaksi :

```
[Program] → [NewsPublisher]  
    |  
    ├── Attach(observer1)  
    ├── Attach(observer2)  
    └── PublishNews() → Notify() → [User.Update()]
```

RINGKASAN :

Class	Peran	Fungsi Utama
IObserver	Interface	Menjamin semua observer punya method Update().
NewsPublisher	Subject/Publisher	Mengelola observer dan mengirim notifikasi.
User	Concrete Observer	Menjalankan aksi saat menerima notifikasi.
Program	Main Program	Mengatur alur eksekusi.



```

using System;
namespace Refactor
{
    class Program
    {
        static void Main(string[] args)
        {
            var sub = new Subject();
            sub.RegisterObserver(new ConcreteObserverA());
            sub.RegisterObserver(new ConcreteObserverB());
            sub.Notify();
            sub.State = 2;
            sub.Notify();
            sub.State = 7;
            sub.Notify();
            sub.State = 1;
            sub.Notify();
            sub.UnregisterObserver(new ConcreteObserverA());
            sub.Notify();
        }
    }
}

Subject: Attached an observer.
Subject: Attached an observer.
Subject: I'm doing something important.
Subject: My state has just changed to: 2
Subject: Notifying observers...
ConcreteObserverA: Reacted to the event.
ConcreteObserverB: Reacted to the event.
Subject: I'm doing something important.
Subject: My state has just changed to: 7
Subject: Notifying observers...
ConcreteObserverB: Reacted to the event.
Subject: Detached an observer.
Subject: I'm doing something important.
Subject: My state has just changed to: 1
Subject: Notifying observers...
ConcreteObserverA: Reacted to the event.
D:\PRAKTIKUM KONTRUKSI PERANGKAT LUNAK\13_Clean_Code_Standard\TPMODUL13_2311104075\tpmodul13_2311104075_N03\tpmodul13_2311104075_N03\bin\Debug\net8.0\tpmodul13_2311104075_N03.exe (process 25012) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

Penjelasan Detail:

Inisialisasi Subject

```
var subject = new Subject();
```

- Membuat objek Subject yang akan menjadi pusat pengamatan. Objek ini akan menyimpan state dan mengelola daftar observer.

Pembuatan Observer A

```
var observerA = new ConcreteObserverA();
```

- ✓ Membuat observer pertama dengan tipe ConcreteObserverA, yang akan bereaksi ketika state subject < 3.

Pendaftaran Observer A

```
subject.Attach(observerA);
```

- ✓ Menambahkan observerA ke daftar observer subject. Sekarang, observerA akan menerima update setiap kali ada perubahan state.

Pembuatan Observer B

```
var observerB = new ConcreteObserverB();
```

- ✓ Membuat observer kedua dengan tipe ConcreteObserverB, yang akan bereaksi ketika state = 0 atau ≥ 2 .

Pendaftaran Observer B

```
subject.Attach(observerB);
```

- Menambahkan observerB ke daftar observer subject. Kedua observer sekarang terdaftar.

Memanggil Business Logic Pertama

```
subject.SomeBusinessLogic();
```

Memicu perubahan state pada subject:

- Generate nilai acak antara 0-10 untuk State.
- Memanggil Notify() untuk memberi tahu semua observer yang terdaftar.
- Output contoh:

- Subject: I'm doing something important.
- Subject: My state has just changed to: 5
- Subject: Notifying observers...
- ConcreteObserverB: Reacted to the event. (karena $5 \geq 2$)

Memanggil Business Logic Kedua

```
subject.SomeBusinessLogic();
```

Proses yang sama diulang dengan nilai state baru. Contoh output:

```
Subject: I'm doing something important.  
Subject: My state has just changed to: 1  
Subject: Notifying observers...  
ConcreteObserverA: Reacted to the event. (karena  $1 < 3$ )  
ConcreteObserverB: Reacted to the event. (karena  $1 \geq 2$ ? Tidak. Tapi  
ada bug di sini!*)
```

Melepaskan Observer B

```
subject.Detach(observerB);
```

- ✓ Menghapus observerB dari daftar observer. Setelah ini, observerB tidak akan menerima notifikasi.

Memanggil Business Logic Ketiga

```
subject.SomeBusinessLogic();
```

Hanya observerA yang akan menerima notifikasi. Contoh output:

```
Subject: I'm doing something important.  
Subject: My state has just changed to: 3
```

Subject: Notifying observers...

(Tidak ada reaksi karena $3 \geq 3$ tidak memicu observerA, dan observerB sudah dilepas)

