

**TUGAS PENDAHULUAN 13**  
**KONSTRUKSI PERANGKAT LUNAK**  
**S1 SOFTWARE ENGINEERING**



**ILHAM LII ASSIDAQ**

**2311104068**

**SE0702**

**DIREKTORAT TELKOM UNIVERSITY PURWOKERTO**

1. Link GitHub : [https://github.com/ilhamlii21/KPL\\_ILHAM\\_LII\\_ASSIDAQ-2311104068\\_S1SE-07-02](https://github.com/ilhamlii21/KPL_ILHAM_LII_ASSIDAQ-2311104068_S1SE-07-02)

2. A. Program.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Threading;
4
5  namespace RefactoringGuru.DesignPatterns.Observer.Conceptual
6  {
7      8 references
8      public interface IObserver
9      {
10         3 references
11         void Update(ISubject subject);
12
13     }
14
15     4 references
16     public interface ISubject
17     {
18         3 references
19         void Attach(IObserver observer);
20
21         // Detach an observer from the subject.
22         2 references
23         void Detach(IObserver observer);
24
25         // Notify all observers about an event.
26         2 references
27         void Notify();
28     }
29
30     // The Subject owns some important state and notifies observers when the
31     // state changes.
32
33     4 references
34     public class Subject : ISubject
35     {
36         // For the sake of simplicity, the Subject's state, essential to all
37         // subscribers, is stored in this variable.
38         5 references
39     }
```

Kode tersebut merupakan implementasi dari design pattern Observer dalam bahasa C#. Pola ini memungkinkan objek Subject untuk memberi tahu sejumlah Observer ketika terjadi perubahan status internalnya. Kelas Subject menyimpan status dalam properti State dan daftar observer-nya dalam \_observers. Saat SomeBusinessLogic() dipanggil, status akan berubah secara acak, dan semua observer yang terdaftar akan diberi tahu melalui metode Notify(). Dua observer konkrit (ConcreteObserverA dan ConcreteObserverB) bereaksi terhadap perubahan status berdasarkan kondisi tertentu. Program ini mencerminkan prinsip pemisahan antara sumber data (subject) dan para pendengar (observers), memungkinkan sistem lebih fleksibel dan modular.

## Running

```
Microsoft Visual Studio Debug Console
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.

C:\Users\ASUS\source\repos\TPMODUL14\TPMODUL14\bin\Debug\net6.0\TPMODUL14.exe (process)
To automatically close the console when debugging stops, enable Tools->Options->Debugging->
Close console when debugging stops.
Press any key to close this window . . .|
```

Output tersebut menunjukkan proses kerja pola Observer saat objek Subject menjalankan logika bisnisnya dan memberitahu para observer. Pertama, dua observer (ConcreteObserverA dan ConcreteObserverB) didaftarkan ke Subject. Saat status berubah menjadi 2, keduanya bereaksi karena memenuhi kondisi masing-masing. Pada perubahan status kedua menjadi 7, hanya ConcreteObserverB yang bereaksi karena ConcreteObserverA hanya merespons jika status kurang dari 3. Setelah ConcreteObserverB dilepas, perubahan status terakhir menjadi 1 hanya memicu reaksi dari ConcreteObserverA. Ini menunjukkan bagaimana observer hanya merespons jika terdaftar dan jika kondisi mereka terpenuhi, mencerminkan fleksibilitas dan efisiensi pola desain Observer.