

TP MODUL 12

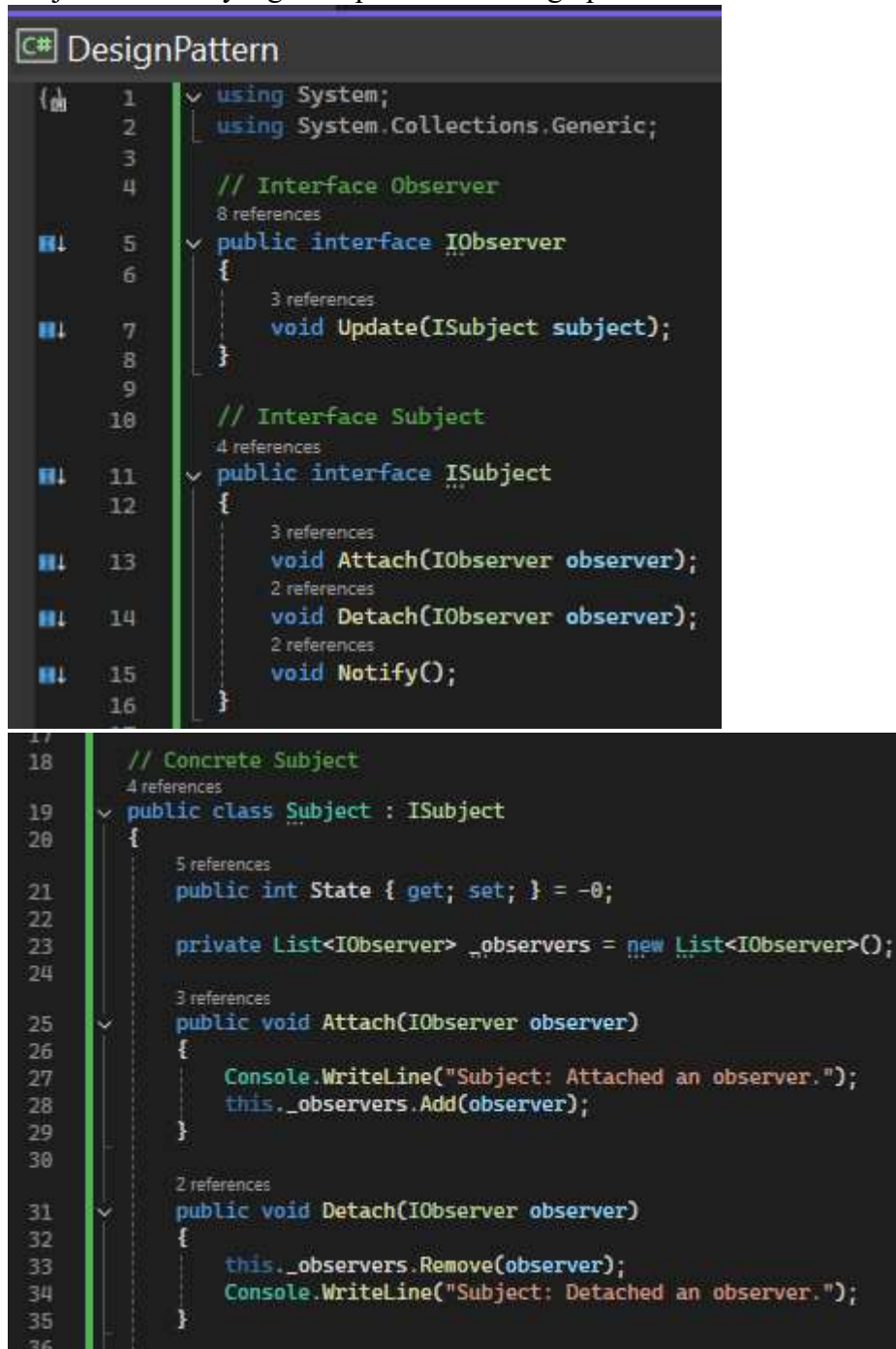
Nama : Farhan Kurniawan

NIM : 2311104073

Kelas : SE-07-02

Link Github : https://github.com/farkurr/KPL_FARHANKURNIAWAN_2311104073_SE-07-02.git

1. Penjelasan Kode yang di implementasi design pattern



```
C# DesignPattern
1  using System;
2  using System.Collections.Generic;
3
4  // Interface Observer
5  public interface IObserver
6  {
7      void Update(ISubject subject);
8  }
9
10 // Interface Subject
11 public interface ISubject
12 {
13     void Attach(IObserver observer);
14     void Detach(IObserver observer);
15     void Notify();
16 }
17
18 // Concrete Subject
19 public class Subject : ISubject
20 {
21     public int State { get; set; } = -0;
22     private List<IObserver> _observers = new List<IObserver>();
23
24     public void Attach(IObserver observer)
25     {
26         Console.WriteLine("Subject: Attached an observer.");
27         this._observers.Add(observer);
28     }
29
30     public void Detach(IObserver observer)
31     {
32         this._observers.Remove(observer);
33         Console.WriteLine("Subject: Detached an observer.");
34     }
35 }
36
```

```

37     2 references
38     public void Notify()
39     {
40         Console.WriteLine("Subject: Notifying observers...");
41         foreach (var observer in _observers)
42         {
43             observer.Update(this);
44         }
45     }
46
47     3 references
48     public void SomeBusinessLogic()
49     {
50         Console.WriteLine("\nSubject: I'm doing something important.");
51         this.State = new Random().Next(0, 10);
52         Console.WriteLine("Subject: My state has just changed to: " + this.State);
53         this.Notify();
54     }
55 }
56

```

```

57 // Concrete Observer A
58 1 reference
59 class ConcreteObserverA : IObserver
60 {
61     2 references
62     public void Update(ISubject subject)
63     {
64         if ((subject as Subject).State < 3)
65         {
66             Console.WriteLine("ConcreteObserverA: Reacted to the event.");
67         }
68     }
69
70 // Concrete Observer B
71 1 reference
72 class ConcreteObserverB : IObserver
73 {
74     2 references
75     public void Update(ISubject subject)
76     {
77         if ((subject as Subject).State == 0 || (subject as Subject).State >= 5)
78         {
79             Console.WriteLine("ConcreteObserverB: Reacted to the event.");
80         }
81     }
82 }
83

```

```

68 // Concrete Observer B
69 1 reference
70 class ConcreteObserverB : IObserver
71 {
72     2 references
73     public void Update(ISubject subject)
74     {
75         if ((subject as Subject).State == 0 || (subject as Subject).State >= 5)
76         {
77             Console.WriteLine("ConcreteObserverB: Reacted to the event.");
78         }
79     }
80 }
81

```

```

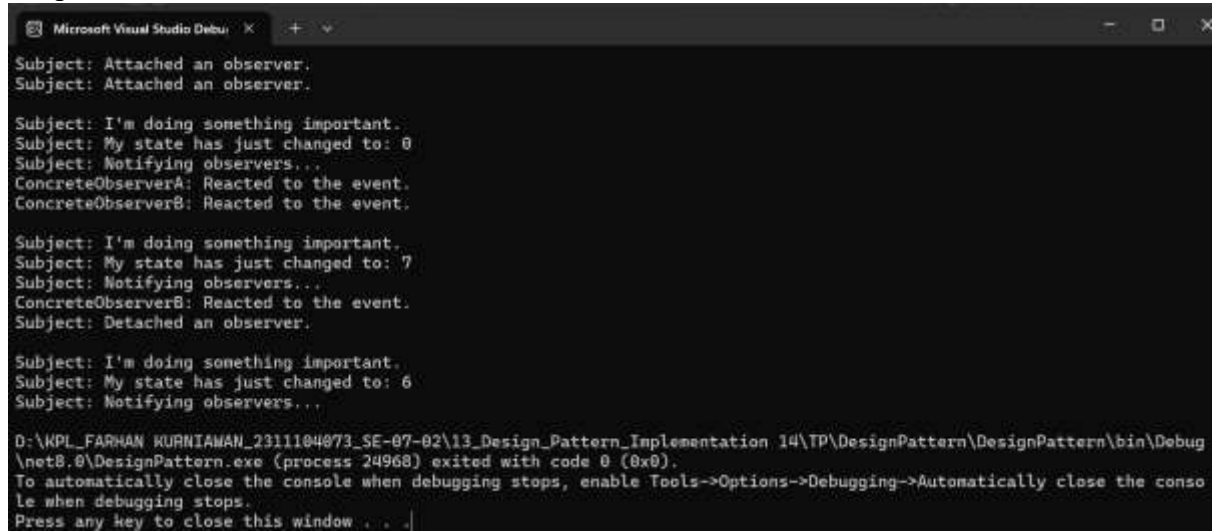
81 // Program Main
82 class Program
83 {
84     static void Main(string[] args)
85     {
86         var subject = new Subject();
87
88         var observerA = new ConcreteObserverA();
89         subject.Attach(observerA);
90
91         var observerB = new ConcreteObserverB();
92         subject.Attach(observerB);
93
94         // Simulasi perubahan state
95         subject.SomeBusinessLogic();
96         subject.SomeBusinessLogic();
97
98         // Melepas salah satu observer
99         subject.Detach(observerB);
100
101         subject.SomeBusinessLogic();
102     }
103 }
104

```

Penjelasan:

Kode tersebut mengimplementasikan Design Pattern Observer, yaitu pola di mana sebuah objek (Subject) dapat memberitahukan perubahan ke sejumlah objek lainnya (Observer) secara otomatis. ISubject adalah interface untuk objek yang diamati, menyediakan method Attach, Detach, dan Notify. Kelas Subject mengimplementasikan interface ini dan menyimpan daftar observer dalam list _observers. Saat method SomeBusinessLogic() dijalankan, Subject mengubah State secara acak, lalu memanggil Notify() untuk memberi tahu semua observer yang telah terdaftar. Interface IObservable mendefinisikan method Update() yang wajib diimplementasikan oleh semua observer. ConcreteObserverA dan ConcreteObserverB adalah dua observer nyata yang akan bereaksi berdasarkan nilai State dari Subject. Fungsi Attach() menambahkan observer, sedangkan Detach() menghapusnya dari daftar. Pola ini berguna saat ada banyak objek yang perlu merespons perubahan dari satu objek pusat tanpa perlu membuat ketergantungan langsung antar objek.

2. Output:



```
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: 0
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: 6
Subject: Notifying observers...

D:\KPL_FARHAN_KURNIAMAN_2311104073_SE-07-02\13_Design_Pattern_Implementation 14\TP\DesignPattern\DesignPattern\bin\Debug\net8.0\DesignPattern.exe (process 24968) 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 . . .
```