

TUGAS PENDAHULUAN
KONSTRUKSI PERANGKAT LUNAK
MODUL 13



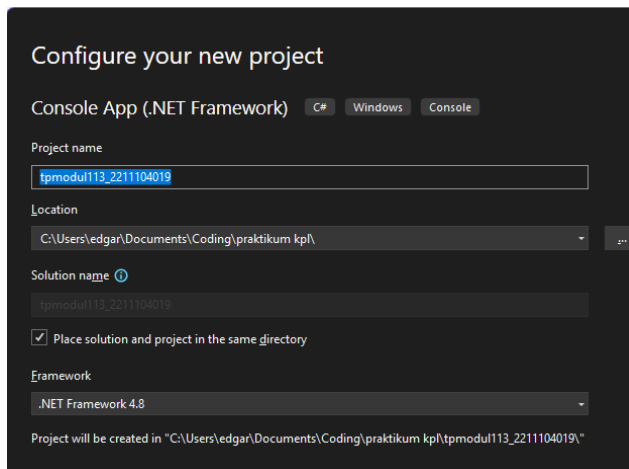
Oleh :

Muhammad Edgar Nadhif
2211104019
SE0601

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

1. Membuat Project Baru

Buat project baru dengan nama tpmodul113_NIM



2. Design Pattern Observer

Design Pattern Observer dapat digunakan dalam aplikasi cuaca, di mana beberapa tampilan seperti suhu dan kelembaban ingin mendapatkan update secara otomatis ketika data cuaca berubah.

3. Langkah-langkah implementasi

- Membuat interface `IObserver` yang memiliki method `Update()` sebagai metode untuk menerima informasi perubahan.

```
namespace tpmodul113_2211104019
{
    8 references
    public interface IObserver
    {
        3 references
        void Update(ISubject subject);
    }
}
```

- Membuat interface `ISubject` yang memiliki method `Attach()`, `Detach()`, dan `Notify()` untuk mengelola daftar observer.

```
namespace tpmodul113_2211104019
{
    4 references
    public interface ISubject
    {
        3 references
        void Attach(IObserver observer);
        2 references
        void Detach(IObserver observer);
        2 references
        void Notify();
    }
}
```

- c. Membuat class Subject yang menyimpan data dan daftar observer, dan memanggil Notify() ketika data berubah.

```
namespace tpmodul113_2211104019
{
    using System;
    using System.Collections.Generic;

    4 references
    public class Subject : ISubject
    {
        5 references
        public int State { get; set; } = 0;
        private List<IObserver> _observers = new List<IObserver>();

        3 references
        public void Attach(IObserver observer)
        {
            Console.WriteLine("Subject: Attached an observer.");
            _observers.Add(observer);
        }

        2 references
        public void Detach(IObserver observer)
        {
            _observers.Remove(observer);
            Console.WriteLine("Subject: Detached an observer.");
        }

        2 references
        public void Notify()
        {
            Console.WriteLine("Subject: Notifying observers...");
            foreach (var observer in _observers)
            {
                observer.Update(this);
            }
        }

        3 references
        public void SomeBusinessLogic()
        {
            Console.WriteLine("\nSubject: I'm doing something important.");
            State = new Random().Next(0, 10);
            Console.WriteLine("Subject: My state has just changed to: " + State);
            Notify();
        }
    }
}
```

- d. Membuat class ConcreteObserver yang mengimplementasikan IObserver untuk merespon perubahan dari Subject.

ConcreteObserverA

```
namespace tpmodul113_2211104019
{
    using System;

    1 reference
    public class ConcreteObserverA : IObserver
    {
        2 references
        public void Update(ISubject subject)
        {
            if ((subject as Subject).State < 3)
            {
                Console.WriteLine("ConcreteObserverA: Reacted to the event.");
            }
        }
    }
}
```

ConcreteObserverB

```
namespace tpmodul113_2211104019
{
    using System;

    1 reference
    public class ConcreteObserverB : IObserver
    {
        2 references
        public void Update(ISubject subject)
        {
            if ((subject as Subject).State == 0 || (subject as Subject).State >= 2)
            {
                Console.WriteLine("ConcreteObserverB: Reacted to the event.");
            }
        }
    }
}
```

- e. Membuat method Main() sebagai program utama yang menjalankan interaksi antara Subject dan Observer.

4. Kelebihan dan Kekurangan Design Pattern Observer

Kelebihan	Kekurangan
Mengurangi ketergantungan antar objek	Jika observer banyak, bisa jadi sulit melacak alur program
Memudahkan update otomatis antar objek	Risiko memory leak jika observer tidak dihapus dengan benar
Observer dapat ditambah atau dihapus secara dinamis	Kompleksitas program bertambah

5. Implementasi Program

Berikut ini adalah potongan kode program yang diimplementasikan di Visual Studio menggunakan C# Console Application:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace tpmodul113_2211104019
{
    using System;

    class Program
    {
        static void Main(string[] args)
        {
            var subject = new Subject();
            var observerA = new ConcreteObserverA();
            subject.Attach(observerA);

            var observerB = new ConcreteObserverB();
            subject.Attach(observerB);

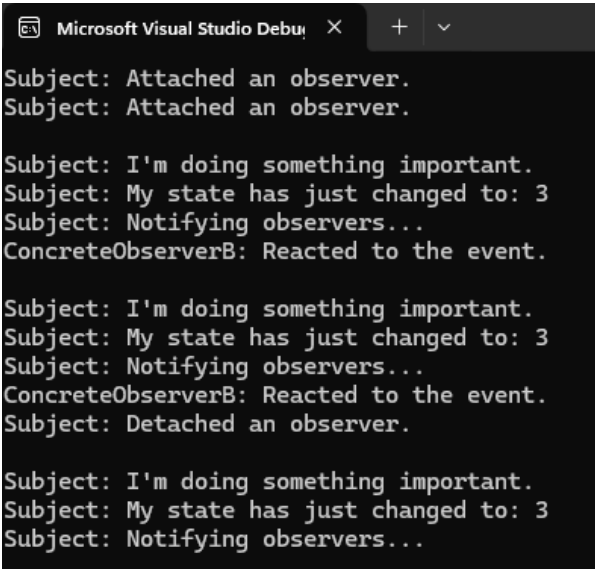
            subject.SomeBusinessLogic();
            subject.SomeBusinessLogic();

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

Penjelasan kode di Main():

- a. `subject.Attach(observerA);` dan `subject.Attach(observerB);` digunakan untuk mendaftarkan observer ke dalam subject supaya menerima pemberitahuan saat data berubah.
- b. `subject.SomeBusinessLogic();` merupakan method yang mengubah data di subject dan kemudian memanggil `Notify()` untuk menginformasikan semua observer yang terdaftar.
- c. `subject.Detach(observerB);` menghapus observer B dari daftar penerima pemberitahuan.

6. Ouput Program



```
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: 3
Subject: Notifying observers...
ConcreteObserverB: Reacted to the event.

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