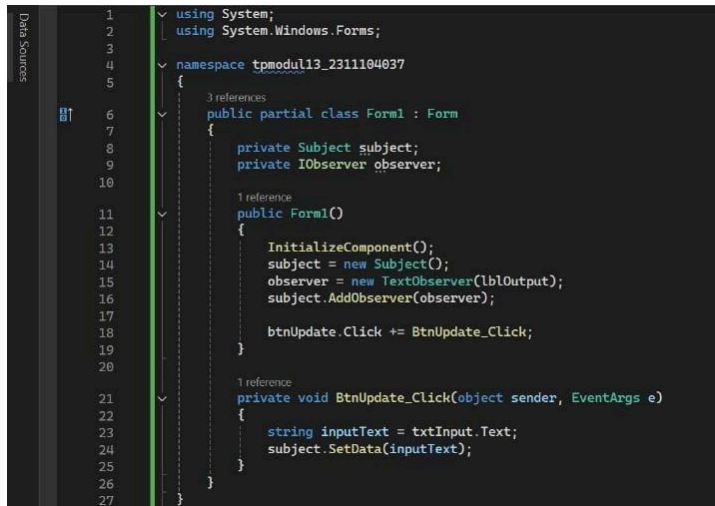


1. Membuat GUI



```
1 using System;
2 using System.Windows.Forms;
3
4 namespace tpmodul13_2311104037
5 {
6     3 references
7     public partial class Form1 : Form
8     {
9         private Subject subject;
10        private IObservable observer;
11
12        1 reference
13        public Form1()
14        {
15            InitializeComponent();
16            subject = new Subject();
17            observer = new TextObserver(lblOutput);
18            subject.AddObserver(observer);
19
20            btnUpdate.Click += BtnUpdate_Click;
21        }
22
23        1 reference
24        private void BtnUpdate_Click(object sender, EventArgs e)
25        {
26            string inputText = txtInput.Text;
27            subject.SetData(inputText);
28        }
29    }
30 }
```

- InitializeComponent(): Menyiapkan elemen GUI seperti TextBox, Button, dan Label.
- Subject dan Observer: Subject menyimpan data dan mengelola observer. TextObserver memantau perubahan data.
- Menambah Observer: Observer didaftarkan ke Subject agar dapat menerima update.
- Event tombol Update: Saat tombol ditekan, teks dari TextBox dikirim ke Subject, lalu Subject memberitahu Observer untuk memperbarui Label dengan teks terbaru.

2. MENJELASKAN SALAH SATU DESIGN PATTERN

- a. Contoh penggunaan design pattern Observer
Pada aplikasi pemantauan cuaca (weather monitoring system), beberapa tampilan GUI (misalnya suhu, kelembaban, tekanan udara) adalah Observer yang menerima update otomatis saat data cuaca di Subject berubah.
- b. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Observer”
 - Buat interface ISubject yang mengatur daftar observer dan method Attach(), Detach(), dan Notify().
 - Buat interface IObservable dengan method Update().
 - Buat kelas Subject yang mengimplementasikan ISubject, menyimpan daftar observer.

- Buat kelas observer yang mengimplementasikan IObserver dan mendefinisikan Update() yang dipanggil oleh Subject saat data berubah.
 - Ketika Subject berubah, panggil Notify() yang memanggil Update() semua observer.
- c. Berikan kelebihan dan kekurangan dari design pattern “Observer”

Kelebihan:

Mendukung loose coupling antar objek, subject dan observer bisa berkembang terpisah.

Sangat cocok untuk aplikasi event-driven, real-time update.

Kekurangan:

Risiko siklus update tak terbatas (jika observer juga mengubah subject).

Bisa terjadi penurunan performa jika terlalu banyak observer.

Sulit debugging saat banyak observer terlibat.

3. IMPLEMENTASI DAN PEMAHAMAN DESIGN PATTERN OBSERVER

Subject menyimpan data dan daftar observer, serta memiliki metode untuk menambah observer dan memberitahu observer jika data berubah.

```

1  using System.Collections.Generic;
2
3  namespace tpmodul13_2311104037
4  {
5      2 references
6      public class Subject
7      {
8          private List<IObserver> observers = new List<IObserver>();
9          private string data;
10
11          1 reference
12          public void AddObserver(IObserver observer)
13          {
14              observers.Add(observer);
15          }
16
17          0 references
18          public void RemoveObserver(IObserver observer)
19          {
20              observers.Remove(observer);
21          }
22
23          1 reference
24          public void SetData(string data)
25          {
26              this.data = data;
27              NotifyObservers();
28          }
29
30          1 reference
31          private void NotifyObservers()
32          {
33          }
34      }
35  }

```

```

28      foreach (var observer in observers)
29      {
30          observer.Update(data);
31      }
32  }
33  }
34  }
35  }

```

Interface IObserver Sebagai kontrak yang harus diikuti oleh setiap observer. Observer wajib memiliki metode Update.

```
1 namespace tpmodul13_2311104037
2 {
3     6 references
4     public interface IObserver
5     {
6         2 references
7         void Update(string data);
8     }
9 }
```

Class TextObserver adalah implementasi dari IObserver, yang akan menampilkan data ke Label saat ada perubahan.

```
1 using System.Windows.Forms;
2
3 namespace tpmodul13_2311104037
4 {
5     2 references
6     public class TextObserver : IObserver
7     {
8         private Label label;
9
10        1 reference
11        public TextObserver(Label label)
12        {
13            this.label = label;
14        }
15
16        2 references
17        public void Update(string data)
18        {
19            label.Text = $"Output: {data}";
20        }
21    }
22 }
```