

Link Github :

https://github.com/MartryatusSofia/KPL_Martryatus-Sofia_2311104003_SE0701/tree/master/13_Design_Pattern/TP_Modul13_2311104003

1. Berikan salah satu contoh kondisi dimana design pattern “Observer” dapat digunakan

Jawab :

Observer pattern bisa digunakan dalam aplikasi notifikasi media sosial. Misalnya, ketika seseorang memposting sesuatu, maka semua follower-nya (observer) akan otomatis diberi tahu.

2. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Observer”

Jawab :

1. Buat interface atau class Observer : objek yang ingin diberi tahu saat terjadi perubahan.
2. Buat Subject (Publisher) : objek yang menyimpan daftar observer dan memberi tahu mereka saat terjadi perubahan.
3. Observer mendaftar ke Subject : dengan metode seperti subscribe() atau attach().
4. Saat Subject berubah, ia memanggil metode update() pada setiap observer yang terdaftar.

3. Berikan kelebihan dan kekurangan dari design pattern “Observer”

4. Jawab :

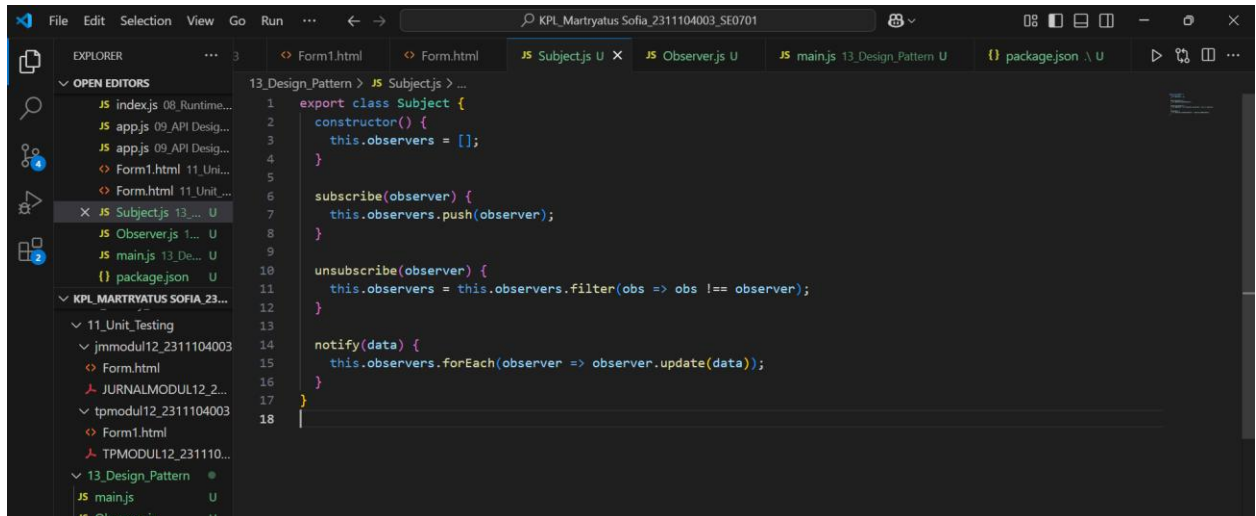
Kelebihan:

- 1) Memisahkan antara subject dan observer, sehingga lebih fleksibel dan mudah diubah.
- 2) Mendukung komunikasi satu-ke-banyak secara otomatis.

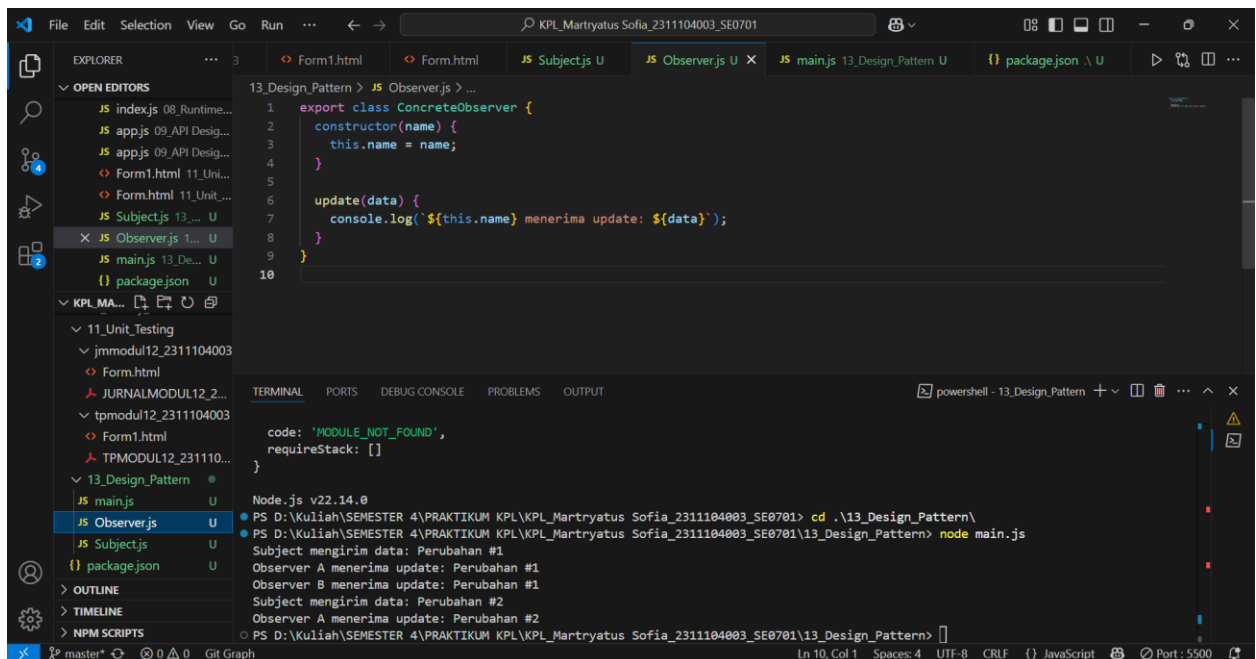
Kekurangan:

- 1) Sulit dilacak jika terdapat terlalu banyak observer.
 - 2) Performa bisa menurun jika terlalu banyak notifikasi dikirim secara bersamaan
5. Pada project yang telah dibuat sebelumnya, tambahkan kode yang mirip atau sama dengan contoh kode yang diberikan di halaman web tersebut

Jawab :



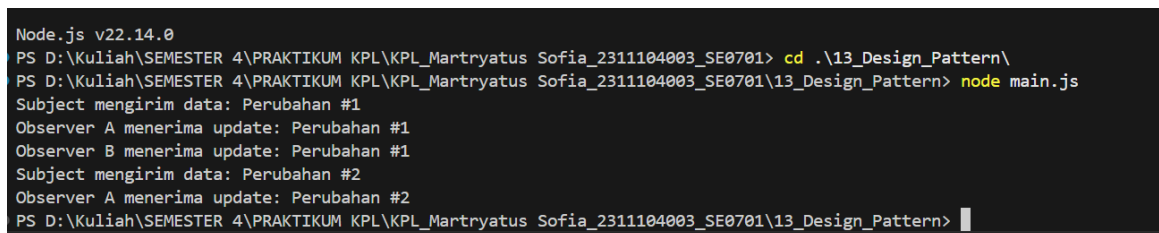
```
1 export class Subject {
2   constructor() {
3     this.observers = [];
4   }
5
6   subscribe(observer) {
7     this.observers.push(observer);
8   }
9
10  unsubscribe(observer) {
11    this.observers = this.observers.filter(obs => obs !== observer);
12  }
13
14  notify(data) {
15    this.observers.forEach(observer => observer.update(data));
16  }
17 }
18
```



```
1 export class ConcreteObserver {
2   constructor(name) {
3     this.name = name;
4   }
5
6   update(data) {
7     console.log(`${this.name} menerima update: ${data}`);
8   }
9 }
10
```

```
code: 'MODULE_NOT_FOUND',
requireStack: []
}
Node.js v22.14.0
PS D:\Kuliah\SEMESTER 4\PRAKTIKUM KPL\KPL_Martryatus Sofia_2311104003_SE0701> cd .\13_Design_Pattern\
PS D:\Kuliah\SEMESTER 4\PRAKTIKUM KPL\KPL_Martryatus Sofia_2311104003_SE0701\13_Design_Pattern> node main.js
Subject mengirim data: Perubahan #1
Observer A menerima update: Perubahan #1
Observer B menerima update: Perubahan #1
Subject mengirim data: Perubahan #2
Observer A menerima update: Perubahan #2
PS D:\Kuliah\SEMESTER 4\PRAKTIKUM KPL\KPL_Martryatus Sofia_2311104003_SE0701\13_Design_Pattern>
```

6. Jalankan program tersebut dan pastikan tidak ada error pada saat project dijalankan



```
Node.js v22.14.0
PS D:\Kuliah\SEMESTER 4\PRAKTIKUM KPL\KPL_Martryatus Sofia_2311104003_SE0701> cd .\13_Design_Pattern\
PS D:\Kuliah\SEMESTER 4\PRAKTIKUM KPL\KPL_Martryatus Sofia_2311104003_SE0701\13_Design_Pattern> node main.js
Subject mengirim data: Perubahan #1
Observer A menerima update: Perubahan #1
Observer B menerima update: Perubahan #1
Subject mengirim data: Perubahan #2
Observer A menerima update: Perubahan #2
PS D:\Kuliah\SEMESTER 4\PRAKTIKUM KPL\KPL_Martryatus Sofia_2311104003_SE0701\13_Design_Pattern>
```

7. Jelaskan tiap baris kode yang terdapat di bagian method utama atau “main”

Jawab :

```
import { Subject } from './Subject.js';  
import { ConcreteObserver } from './Observer.js';
```

Import modul Subject dan ConcreteObserver dari file lain (Subject.js dan Observer.js). Ini digunakan agar file main.js bisa menggunakan class-class yang sudah dibuat di file terpisah tersebut.

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

Membuat **instance** dari Subject, yaitu objek yang bisa memberi notifikasi ke para observer ketika terjadi perubahan data.

```
const observer1 = new ConcreteObserver("Observer A");  
const observer2 = new ConcreteObserver("Observer B");
```

Membuat dua observer (pengamat), masing-masing diberi nama "Observer A" dan "Observer B". Mereka akan menerima notifikasi dari subject.

```
subject.subscribe(observer1);  
subject.subscribe(observer2);
```

Mendaftarkan kedua observer ke dalam subject. Artinya, observer1 dan observer2 akan menerima update setiap kali subject memberi notifikasi.

```
console.log("Subject mengirim data: Perubahan #1");  
subject.notify("Perubahan #1");
```

Menampilkan pesan di console, lalu subject mengirimkan notifikasi ke semua observer yang terdaftar dengan data "Perubahan #1". Observer yang terdaftar akan menerima dan menampilkan pesan bahwa mereka menerima update tersebut.

```
subject.unsubscribe(observer2);
```

Menghapus observer2 dari daftar observer. Setelah ini, observer2 tidak akan menerima notifikasi lagi.

```
console.log("Subject mengirim data: Perubahan #2");  
subject.notify("Perubahan #2");
```

Menampilkan pesan di console, lalu subject kembali mengirimkan notifikasi, kali ini hanya ke observer1 karena observer2 sudah di-unsubscribe.