TP13_2311104002_Shilfi Habibah_SE0701

I. Link github
https://github.com/shilfihabibah/KPL_Shilfi-Habibah_2311104002_SE07-01/tree/master/13_Design_Pattern

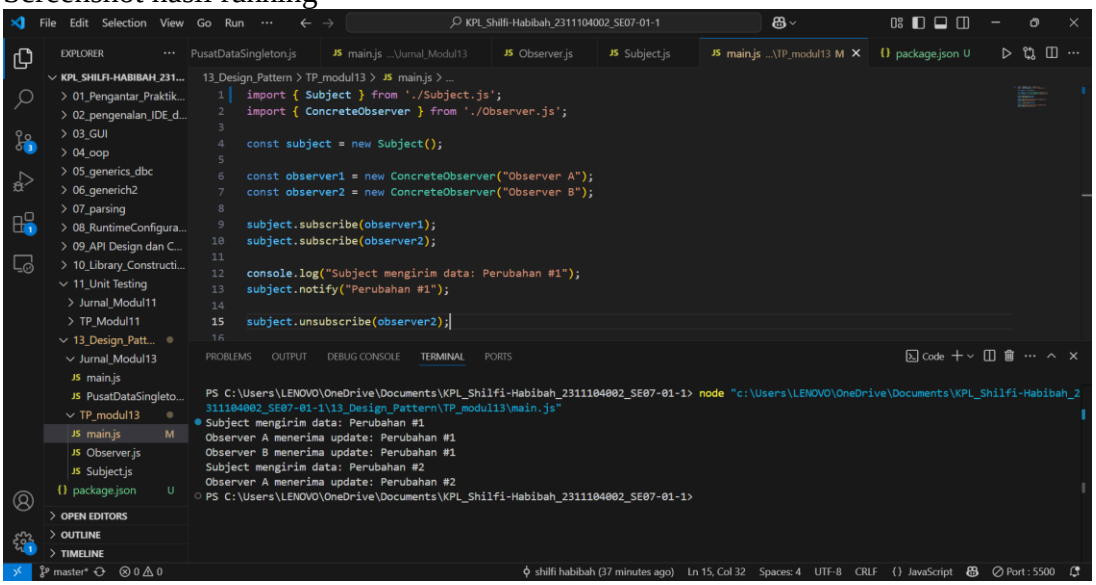
II. Pertanyaan

1. Berikan salah satu contoh kondisi dimana design pattern “Observer” dapat digunakan
   a. Manajemen Koneksi Database
      Ketika sebuah aplikasi hanya perlu satu koneksi ke database untuk efisiensi dan konsistensi, Singleton digunakan agar semua bagian aplikasi menggunakan koneksi yang sama.
   b. Logger (Pencatatan Log)
      Aplikasi yang membutuhkan pencatatan log biasanya hanya menggunakan satu instance logger agar log tidak tercampur atau tumpang tindih.
2. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Singleton”.
   1. Buat Constructor (private / terbatas)
      Hindari agar objek tidak bisa dibuat langsung menggunakan new.
   2. Buat Static Property untuk Menyimpan Instance
      Misalnya: static _instance = null.
   3. Buat Static Method getInstance()
      Method ini akan:
      - Mengecek apakah instance sudah ada.
      - Jika belum, membuat instance baru.
      - Jika sudah ada, mengembalikan instance yang sama.
   4. Gunakan Hanya Melalui getInstance()
      Semua akses ke Singleton harus dilakukan melalui method ini.
3. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.
   Kelebihan:
   a) Menghemat Memori
      Hanya satu objek dibuat dan digunakan bersama.
   b) Konsistensi Data
      Semua bagian program bekerja dengan instance yang sama.
   c) Pengendalian Akses Global
      Mudah mengontrol sumber daya dari satu titik.

   Kekurangan:
   a) Sulit Diuji (Testing)
      Karena sifat global-nya, sulit untuk membuat tiruan (mock) saat unit testing.
   b) Pelanggaran Prinsip OOP (Single Responsibility)
      Singleton bisa menjadi "god object" karena digunakan di banyak tempat.

c) Masalah di Lingkungan Multithread
Tanpa penanganan yang benar, bisa terjadi masalah ketika banyak thread mencoba membuat instance bersamaan (di beberapa bahasa).

III. Screenshot hasil running



IV. Codingan

Subject.js

```
13_Design_Pattern > TP_modul13 > JS main.js > ...
1  import { Subject } from './Subject.js';
2  import { ConcreteObserver } from './Observer.js';
3
4  const subject = new Subject();
5
6  const observer1 = new ConcreteObserver("Observer A");
7  const observer2 = new ConcreteObserver("Observer B");
8
9  subject.subscribe(observer1);
10 subject.subscribe(observer2);
11
12 console.log("Subject mengirim data: Perubahan #1");
13 subject.notify("Perubahan #1");
14
15 subject.unsubscribe(observer2);
16
17 console.log("Subject mengirim data: Perubahan #2");
18 subject.notify("Perubahan #2");
```

a) Menyimpan daftar observer.
b) Punya fungsi:
- subscribe(observer): menambahkan observer.
- Untuk unsubscribe(observer): menghapus observer dari daftar.
- notify(data): mengirim data ke semua observer yang terdaftar.

Observer.js

13_Design_Pattern > TP_modul13 > JS Observer.js > ConcreteObserver > update

```
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  }
```

a) Mewakili observer nyata.
b) Mencetak ke konsol saat menerima data dari subject melalui method update(data).

Main.js

13_Design_Pattern > Jurnal_Modul13 > JS main.js > ...

```
1  import { PusatDataSingleton } from './PusatDataSingleton.js';
2
3  try {
4    const pusat1 = PusatDataSingleton.getInstance();
5    pusat1.tambahData("Data 1");
6    pusat1.tambahData("Data 2");
7
8    const pusat2 = PusatDataSingleton.getInstance();
9    pusat2.tambahData("Data 3");
10
11   console.log("Semua Data dari pusat1:", pusat1.getSemuaData());
12   console.log("Semua Data dari pusat2:", pusat2.getSemuaData());
13
14   console.log("Apakah pusat1 === pusat2?", pusat1 === pusat2);
15 } catch (err) {
16   console.error(err.message);
17 }
```

a) Membuat instance dari Subject.
b) Membuat dua observer: Observer A dan Observer B.
c) Mendaftarkan keduanya ke subject.
d) Mengirim notifikasi "Perubahan #1" ke semua observer.
e) Observer B dihapus dari daftar observer.
f) Mengirim notifikasi "Perubahan #2" hanya ke Observer A.