Jawaban soal untuk TP

A. Berikan salah satu contoh kondisi dimana design pattern “Observer” dapat digunakan
Observer pattern cocok digunakan ketika ada satu objek (subject) yang berubah dan perubahan tersebut perlu diberitahukan kepada satu atau banyak objek lainnya (observers) secara otomatis.
Contoh:
Sistem notifikasi pada media sosial.
Misalnya, saat seorang pengguna (subject) mengunggah postingan baru, semua pengikutnya (observers) akan mendapatkan notifikasi secara otomatis.
B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Observer”
Langkah langkahnya yakni sebagai berikut ini:
- membuat interface observer: Mendefinisikan metode update() yang akan dipanggil saat terjadi perubahan pada subject.
- membuat interface subjek: definisi metod untuk Menambahkan observer (attach()), Menghapus observer (detach()), dan Memberi notifikasi kepada observer (notifyObservers()).
- Implementasikan Kelas Subject Konkrit: Kelas ini menyimpan daftar observers dan memanggil update() milik observer saat ada perubahan.
- Implementasikan Kelas Observer Konkrit: Kelas ini akan melakukan aksi tertentu ketika menerima notifikasi dari subject.
- Hubungkan Observer dengan Subject Daftarkan observer ke subject agar bisa menerima notifikasi saat terjadi perubahan.
C. Berikan kelebihan dan kekurangan dari design pattern “Observer”
Kelebihan:
- Loose coupling: Subject tidak perlu tahu detail implementasi observer, cukup menggunakan interface.
- Scalability: Mudah menambah observer baru tanpa mengubah subject.
- Real-time update: Perubahan pada subject langsung disampaikan ke semua observer.
Kekurangan:
- Potensi overhead: Jika jumlah observer sangat banyak, proses notifikasi bisa memakan banyak waktu.
- Urutan eksekusi tidak terjamin: Tidak dapat memastikan observer mana yang akan dipanggil lebih dulu.
- Kesulitan debug: Karena notifikasi terjadi secara otomatis, bisa membuat alur program sulit dilacak.

Implementasi TP

1. File observer.js – Implementasi Design Pattern Observer

Pada file ini dibuat dua kelas utama yang mencerminkan konsep **Observer Pattern**:

Class Subject

- Bertugas sebagai **pusat notifikasi** yang mengelola sekumpulan observer.
- Memiliki properti:
  - observers: array untuk menyimpan daftar observer yang sudah didaftarkan.
  - state: nilai yang bisa berubah dan akan diberitahukan ke semua observer.
- Metode:
  - attach(observer): untuk mendaftarkan observer baru.
  - detach(observer): untuk menghapus observer tertentu.
  - notify(): memanggil method update() pada semua observer saat state berubah.
  - setState(newState): mengubah state dan langsung men-trigger notify().

Class ConcreteObserver

- Mewakili **komponen yang ingin menerima notifikasi**.
- Memiliki properti nama untuk identifikasi.
- Memiliki method update(state) untuk menerima data terbaru dari Subject.

2. File main.js – Testing Observer Pattern

File ini berfungsi sebagai **program utama** yang menguji implementasi Observer Pattern:

1. Pertama, dua instance dari ConcreteObserver dibuat dengan nama "Christian" dan "Felix".
2. Observer tersebut di-*register* ke dalam subject menggunakan attach().
3. Saat setState() dipanggil, semua observer yang terdaftar akan menerima notifikasi melalui update().
4. Kemudian, observer "Felix" dihapus dengan detach().
5. Saat setState() dipanggil lagi, hanya "Christian" yang menerima update.

```
jerry@Lumia MINGW64 ~/OneDrive/Documents/Belajar koding/KP
$ node main.js
Subject: State berubah ke "Praktikum Modul 13 dimulai!"
Christian menerima update: Praktikum Modul 13 dimulai!
Felix menerima update: Praktikum Modul 13 dimulai!
Subject: State berubah ke "Modul 13 selesai dikerjakan!"
Christian menerima update: Modul 13 selesai dikerjakan!
```

Jawaban untuk soal Jurnal

A. Berikan salah dua contoh kondisi dimana design pattern “Singleton” dapat digunakan.

1. Manajemen Koneksi Database

Dalam aplikasi, hanya satu objek koneksi database yang dibutuhkan agar efisien dan konsisten. Singleton memastikan hanya ada satu instance koneksi yang digunakan bersama oleh seluruh aplikasi.

2. Logging System (Sistem Pencatatan Log)

Logger biasanya hanya memerlukan satu instance yang digunakan di seluruh sistem untuk mencatat aktivitas. Dengan Singleton, semua bagian sistem mencatat ke tempat yang sama tanpa membuat logger baru setiap kali.

B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Singleton”.

1. Buat Kelas dengan Konstruktor Privat: Agar objek tidak bisa diinstansiasi dari luar kelas.
2. Buat Variabel Statis untuk Menyimpan Instance: Variabel ini akan menyimpan satu-satunya instance dari kelas tersebut.
3. Buat Method Publik Statis untuk Mengakses Instance: Method ini akan memeriksa apakah instance sudah dibuat. Jika belum, maka akan dibuat dan dikembalikan. Jika sudah, langsung mengembalikan instance yang ada.

Contoh psuedo codenya yaitu:

```
public class Singleton {
   private static Singleton instance;

   private Singleton() {
      // konstruktor privat
   }

   public static Singleton getInstance() {
      if (instance == null) {
         instance = new Singleton();
      }
      return instance;
   }
}
```

C. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.

Kelebihan:

1. Kontrol Akses Global ke Objek: Memberikan titik akses tunggal (global access point) ke objek.
2. Menghemat Memori dan Resource: Karena hanya satu instance yang dibuat selama runtime.
3. Konsistensi Data: Cocok untuk objek yang menyimpan state global atau konfigurasi yang konsisten di seluruh aplikasi.

Kekurangan:

1. Susah untuk Diuji (Testing): Karena instance bersifat global dan tidak mudah diubah, sulit dilakukan mocking saat pengujian.
2. Tidak Fleksibel untuk Perluasan: Sulit untuk mewariskan class Singleton karena konstruktor bersifat privat.
3. Potensi Masalah di Lingkungan Multithread: Jika tidak diimplementasikan dengan benar (misalnya tanpa sinkronisasi), dapat menyebabkan race condition saat membuat instance.

Implementasi Jurnal

1. Class PusatDataSingleton (File pusatDataSingleton.js)

Kelas PusatDataSingleton merupakan representasi dari **design pattern Singleton**, yaitu pola desain yang memastikan **hanya ada satu instance** dari suatu kelas yang digunakan di seluruh program.

Penjelasan Elemen Penting:

- DataTersimpan: array yang menyimpan semua data string yang ditambahkan.
- _instance: properti statis yang menyimpan satu-satunya objek singleton.
- Constructor (constructor()):
    - Jika _instance sudah ada, constructor mengembalikan objek tersebut.
    - Jika belum ada, objek baru dibuat dan disimpan di _instance.
- GetDataSingleton():
    - Merupakan metode statis yang memastikan hanya satu instance dibuat.
- AddSebuahData(input):
    - Menambahkan string baru ke list DataTersimpan.
- HapusSebuahData(index):
    - Menghapus data dari list berdasarkan indeks.
- PrintSemuaData():
    - Menampilkan isi data satu per satu ke console.
- GetSemuaData():
    - Mengembalikan seluruh array data.

Dengan desain ini, seluruh akses ke data dilakukan melalui instance yang **selalu sama**, tidak peduli dari variabel mana kita mengaksesnya.

2. Testing di File main.js

File main.js digunakan untuk menguji apakah class PusatDataSingleton benar-benar menerapkan konsep singleton.

Langkah-langkah:

1. **Ambil Instance**:
    - Dua variabel data1 dan data2 diisi dengan hasil dari GetDataSingleton().
    - Karena ini singleton, data1 dan data2 seharusnya menunjuk ke **objek yang sama**.
2. **Tambah Data**:
    - Melalui data1, ditambahkan beberapa data nama: "Christian Felix", "Revan Kurniawan", dan "Fazza Abiyu".
3. **Cetak Data dari data2**:
    - PrintSemuaData() digunakan untuk mencetak isi dari singleton lewat data2.
    - Harusnya output-nya menampilkan semua data yang ditambahkan lewat data1.
4. **Hapus Data Lewat data2**:
    - Salah satu nama asisten dihapus menggunakan HapusSebuahData().
5. **Cetak Ulang lewat data1**:
    - Menunjukkan bahwa perubahan di data2 juga berpengaruh ke data1, karena mereka satu objek yang sama.
6. **Hitung Jumlah Data**:
    - Digunakan GetSemuaData().length untuk mengecek bahwa jumlah elemen di data1 dan data2 tetap konsisten.

Hasil running kodingan