**Tugas Jurnal Modul 13**

1. Penjelasan Design Pattern Singleton

**A. Dua Situasi Umum Penggunaan Pola Desain Singleton**

1. **Logger (Pencatat Aktivitas Aplikasi)**

Dalam sistem logging, biasanya hanya dibutuhkan satu instance logger agar semua bagian aplikasi mencatat informasi ke sumber yang sama. Hal ini mencegah duplikasi file log dan menjaga keseragaman dalam pencatatan data.

1. **Koneksi ke Database**

Dalam mengelola koneksi database, sering kali hanya satu koneksi utama yang digunakan dan dibagikan ke seluruh bagian aplikasi. Dengan menggunakan pola Singleton, hanya satu objek koneksi dibuat, yang membantu mengurangi penggunaan sumber daya dan mencegah konflik dalam akses data.

1. **Langkah-Langkah Implementasi Pola Desain Singleton**
   * Jadikan konstruktor kelas bersifat private atau batasi akses langsung terhadapnya (di JavaScript, ini biasanya dilakukan dengan memeriksa apakah instance sudah dibuat karena tidak mendukung konstruktor privat secara langsung).
   * Definisikan atribut statis dalam kelas untuk menyimpan satu-satunya instance.
   * Buat method statis seperti getInstance() yang bertanggung jawab mengembalikan instance. Jika instance belum dibuat, method ini akan membuatnya terlebih dahulu.
   * Semua akses terhadap instance kelas harus melalui method tersebut, bukan dengan membuat instance baru menggunakan new.

1. **Kelebihan dan Kekurangan Pola Singleton** **Kelebihan:**
   * **Kontrol Terpusat**: Memastikan hanya satu instance yang digunakan secara global di seluruh aplikasi.
   * **Efisiensi Penggunaan Sumber Daya**: Sangat berguna untuk komponen yang mahal seperti koneksi database atau konfigurasi umum.
   * **Akses Mudah**: Dapat digunakan dari bagian manapun dalam program tanpa harus mendistribusikan objek secara eksplisit.

**Kekurangan:**

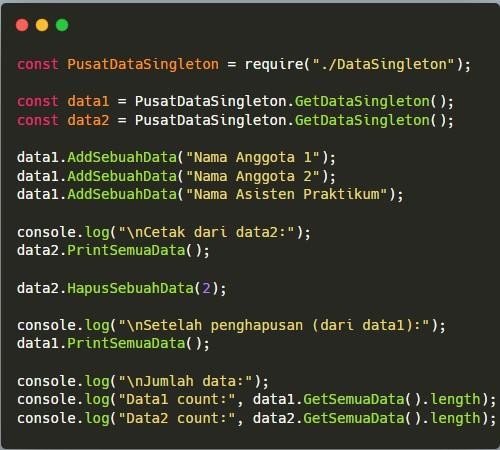
* + **Sulit dalam Pengujian**: Karena bersifat global, sulit mengganti atau mengatur ulang instance selama pengujian unit.
  + **Melanggar Prinsip Tanggung Jawab Tunggal**: Kelas bertanggung jawab atas logika bisnis sekaligus pengelolaan instansinya sendiri.
  + **Ketergantungan Tersembunyi**: Karena bisa langsung diakses, ketergantungan antar komponen jadi tidak terlihat, menyulitkan pemeliharaan dan pengembangan lebih lanjut

1. DataSingleton.js



Class PusatDataSingleton adalah implementasi dari pola Singleton yang memastikan hanya satu instance dari class ini yang digunakan di seluruh aplikasi. Instance disimpan dalam properti statis \_instance, dan hanya dibuat sekali melalui method GetDataSingleton(). Class ini menyimpan data dalam array DataTersimpan, serta menyediakan method untuk menambahkan, menghapus, dan menampilkan data. Dengan cara ini, semua perubahan data akan bersifat global karena setiap akses menuju objek Singleton yang sama.

1. Main.js



Kode di atas merupakan implementasi penggunaan class PusatDataSingleton dengan pola Singleton. Dua variabel (data1 dan data2) diisi menggunakan method GetDataSingleton(), yang memastikan keduanya mengacu pada instance yang sama. Melalui data1, ditambahkan tiga data, lalu isi datanya ditampilkan menggunakan data2. Setelah itu, salah satu data (yaitu nama asisten praktikum) dihapus melalui data2, dan hasilnya kembali dicetak melalui data1. Karena keduanya adalah instance yang sama, perubahan yang dilakukan di data2 juga terlihat di data1, membuktikan bahwa pola Singleton berhasil diterapkan.

**Tugas Pendahuluan Modul 13**

1. Penjelasan Design Pattern Observer A. Contoh Kondisi

Penggunaan Observer

Observer pattern cocok digunakan pada sistem **notifikasi**. Misalnya, dalam aplikasi cuaca, setiap kali data cuaca berubah, semua komponen yang menampilkan informasi (seperti suhu, kelembapan, dll) harus diperbarui secara otomatis. Observer memudahkan banyak *subscriber* untuk mengikuti perubahan tanpa mengganggu logika utama.

* 1. Langkah-langkah Implementasi Observer
     1. Buat class **Subject** yang menyimpan daftar observer dan menyediakan method untuk menambah, menghapus, dan memberi tahu observer.
     2. Buat class **Observer** yang memiliki method update() untuk menerima notifikasi.
     3. Saat state Subject berubah, ia akan memanggil method update() milik semua observer yang terdaftar.
  2. Kelebihan dan Kekurangan Observer **Kelebihan:**
     1. Memisahkan logika antara objek utama (subject) dan pengamat (observer). 2.

Mempermudah penambahan pengamat baru tanpa mengubah kode subject.

3. Cocok untuk event-driven systems.

**Kekurangan:**

* + 1. Sulit dilacak saat jumlah observer banyak (debugging).
    2. Performa menurun jika terlalu banyak observer aktif.
    3. Ketergantungan tersembunyi antara subject dan observer jika tidak didokumentasikan dengan baik.

1. ObserverPattern.js



Kode di atas merupakan implementasi pola Observer dalam JavaScript, di mana class Subject berperan sebagai objek yang diamati dan menyimpan daftar observer. Method attach() digunakan untuk menambahkan observer, sedangkan detach() untuk menghapusnya. Ketika state berubah melalui setState(), method notify() akan dipanggil untuk memberi tahu seluruh observer yang masih aktif. Class ConcreteObserver mewakili objek pengamat yang akan merespons perubahan dengan mencetak status terbaru. Dalam program utama, dua observer ditambahkan, lalu state diubah beberapa kali untuk menunjukkan bagaimana observer menerima notifikasi, dan setelah observer1 dilepas, hanya observer2 yang menerima notifikasi perubahan selanjutnya.

Hasil running :

