Nama : Naufal Aflakh Wijayanto

NIM : 2211104073

Kelas : SE063

MENJELASKAN DESIGN PATTERN SINGLETON

- **Kondisi di Mana Singleton Dapat Digunakan**:
  - **Contoh 1**: Penggunaan dalam koneksi ke database. Ketika aplikasi membutuhkan koneksi database yang tunggal agar tidak ada banyak koneksi yang tercipta.
  - **Contoh 2**: Penggunaan dalam pengelolaan konfigurasi aplikasi. Singleton memastikan hanya ada satu instance dari pengaturan aplikasi yang digunakan di seluruh aplikasi.
- **Langkah-langkah Mengimplementasikan Design Pattern Singleton**:
  - **Langkah 1**: Membuat class dengan constructor privat yang memastikan bahwa hanya satu instance yang dapat dibuat.
  - **Langkah 2**: Menyediakan method statis yang dapat diakses untuk mendapatkan instance tunggal tersebut.
  - **Langkah 3**: Jika instance sudah ada, kembalikan instance tersebut; jika belum, buatlah instance baru.
- **Kelebihan dan Kekurangan Singleton**:
  - **Kelebihan**:
    - Menjamin hanya ada satu instance dari sebuah objek.
    - Memungkinkan kontrol yang mudah terhadap akses instance tunggal tersebut.
    - Mempermudah pengelolaan sumber daya yang terbatas seperti koneksi ke database.
  - **Kekurangan**:
    - Dapat menyebabkan kesulitan dalam pengujian unit karena kesulitan dalam memanipulasi status instance tunggal.
    - Membatasi fleksibilitas karena instance hanya ada satu.
    - Tidak mudah diimplementasikan dalam lingkungan multithreaded jika tidak ditangani dengan baik.

Implementasi dan Pemahaman Design Pattern Singleton

Membuat Class Singleton "PusatDataSingleton" di script.js

Script.js

```
class PusatDataSingleton {
  // Atribut untuk menyimpan data
  constructor() {
    if (PusatDataSingleton._instance) {
      return PusatDataSingleton._instance; // Mengembalikan instance yang sudah ada
    }
    this.DataTersimpan = []; // List untuk menyimpan data
    PusatDataSingleton._instance = this; // Menyimpan instance tunggal
  }

  // Method untuk mendapatkan instance tunggal
  static GetDataSingleton() {
    if (!PusatDataSingleton._instance) {
      new PusatDataSingleton(); // Membuat instance pertama jika belum ada
    }
    return PusatDataSingleton._instance; // Mengembalikan instance tunggal
  }

  // Method untuk mengambil semua data
  GetSemuaData() {
    return this.DataTersimpan;
  }

  // Method untuk menambahkan data baru
  AddSebuahData(input) {
    this.DataTersimpan.push(input);
  }

  // Method untuk menghapus data berdasarkan index
  HapusSebuahData(index) {
    this.DataTersimpan.splice(index, 1);
  }

  // Method untuk mencetak semua data
  PrintSemuaData() {
    this.DataTersimpan.forEach(data => {
      console.log(data);
    });
  }
```

```
}
```

Kode JavaScript di atas merupakan implementasi dari *design pattern* **Singleton**, yang bertujuan memastikan hanya ada **satu objek instance** dari suatu kelas selama siklus hidup program. Kelas PusatDataSingleton menyimpan data dalam array DataTersimpan, dan menyediakan berbagai method untuk mengelola data: AddSebuahData() untuk menambahkan item, HapusSebuahData() untuk menghapus berdasarkan indeks, GetSemuaData() untuk mengambil semua data, serta PrintSemuaData() untuk mencetak seluruh isi array ke konsol. Fungsi GetDataSingleton() digunakan untuk mendapatkan instance tunggal dari kelas ini. Jika instance belum ada, maka dibuat baru; jika sudah ada, akan dikembalikan instance yang sama. Pola ini sangat berguna untuk pengelolaan data terpusat yang konsisten dalam skala aplikasi besar.

Implementasi Program Utama

Menambahkan Implementasi di Program Utama (script.js)

```
class PusatDataSingleton {
  // Atribut untuk menyimpan data
  constructor() {
    if (PusatDataSingleton._instance) {
      return PusatDataSingleton._instance; // Mengembalikan instance yang sudah ada
    }
    this.DataTersimpan = []; // List untuk menyimpan data
    PusatDataSingleton._instance = this; // Menyimpan instance tunggal
  }

  // Method untuk mendapatkan instance tunggal
  static GetDataSingleton() {
    if (!PusatDataSingleton._instance) {
      new PusatDataSingleton(); // Membuat instance pertama jika belum ada
    }
    return PusatDataSingleton._instance; // Mengembalikan instance tunggal
  }

  // Method untuk mengambil semua data
  GetSemuaData() {
    return this.DataTersimpan;
  }

  // Method untuk menambahkan data baru
```

```
  AddSebuahData(input) {
    this.DataTersimpan.push(input);
  }

  // Method untuk menghapus data berdasarkan index
  HapusSebuahData(index) {
    this.DataTersimpan.splice(index, 1);
  }

  // Method untuk mencetak semua data
  PrintSemuaData() {
    return this.DataTersimpan.join('<br>'); // Menggunakan <br> agar data dipisahkan baris
baru
  }
}

// Implementasi Program Utama

// Fungsi untuk menambah data
function tambahData() {
  const data1 = PusatDataSingleton.GetDataSingleton();
  data1.AddSebuahData("Anggota 1");
  data1.AddSebuahData("Anggota 2");
  data1.AddSebuahData("Asisten Praktikum");

  // Menambahkan data ke data2 dan menampilkan semua data
  const data2 = PusatDataSingleton.GetDataSingleton();
  tampilkanData();  // Menampilkan data yang sudah ditambahkan
}

// Fungsi untuk menampilkan semua data
function tampilkanData() {
  const data2 = PusatDataSingleton.GetDataSingleton();
  const dataContainer = document.getElementById('dataContainer');
  dataContainer.innerHTML = ''; // Menghapus data sebelumnya

  // Menampilkan semua data di halaman web
  const data = data2.PrintSemuaData();
  dataContainer.innerHTML = data; // Memasukkan data ke dalam elemen dataContainer
}

// Fungsi untuk menghapus data
function hapusData() {
  const data2 = PusatDataSingleton.GetDataSingleton();
```

```
  data2.HapusSebuahData(2); // Menghapus "Asisten Praktikum"
  tampilkanData(); // Menampilkan data setelah penghapusan
}
```

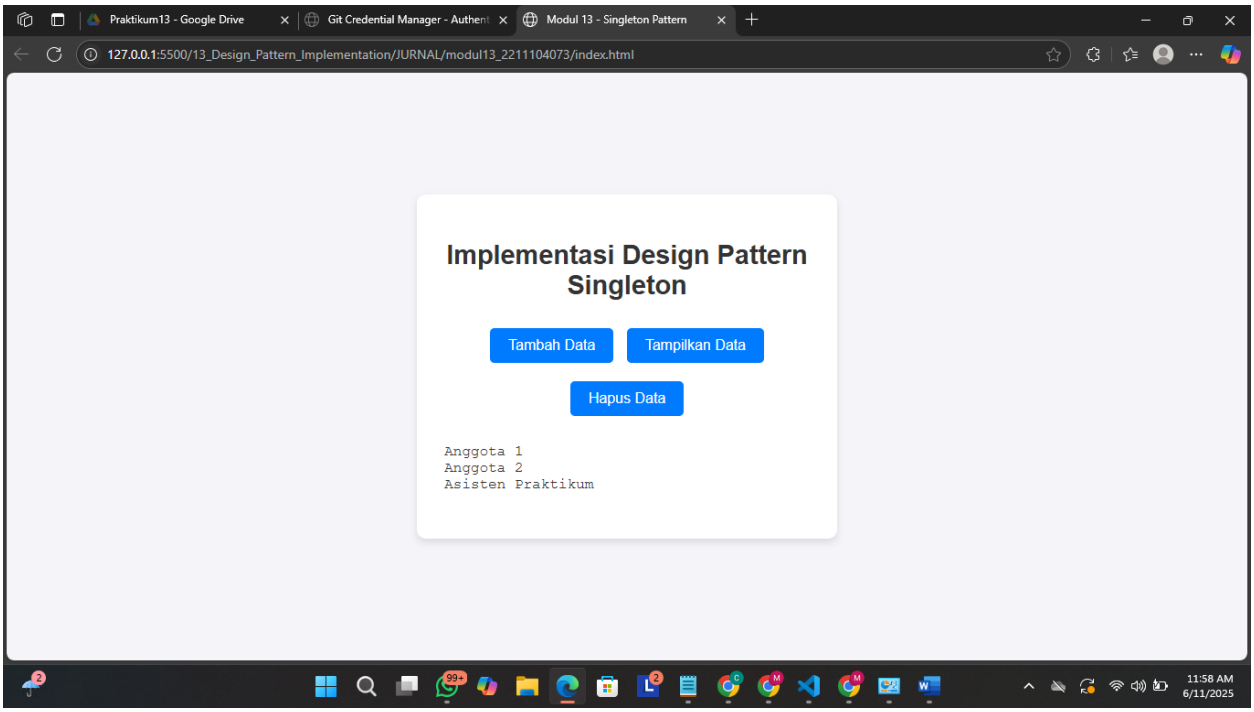
Kode di atas merupakan contoh implementasi **pola Singleton** yang terintegrasi dengan tampilan web (HTML) untuk mengelola dan menampilkan data secara terpusat. Kelas PusatDataSingleton hanya dapat memiliki satu instance yang menyimpan semua data dalam array DataTersimpan. Fungsi tambahData() digunakan untuk menambahkan data baru ke array, sedangkan tampilkanData() akan menampilkan isi data tersebut ke halaman web melalui elemen HTML dengan ID dataContainer. Fungsi hapusData() berfungsi untuk menghapus data pada indeks tertentu, lalu memperbarui tampilan dengan memanggil kembali tampilkanData(). Dengan pendekatan ini, semua interaksi data tetap konsisten karena dikelola dari satu sumber yang sama, yaitu instance Singleton.

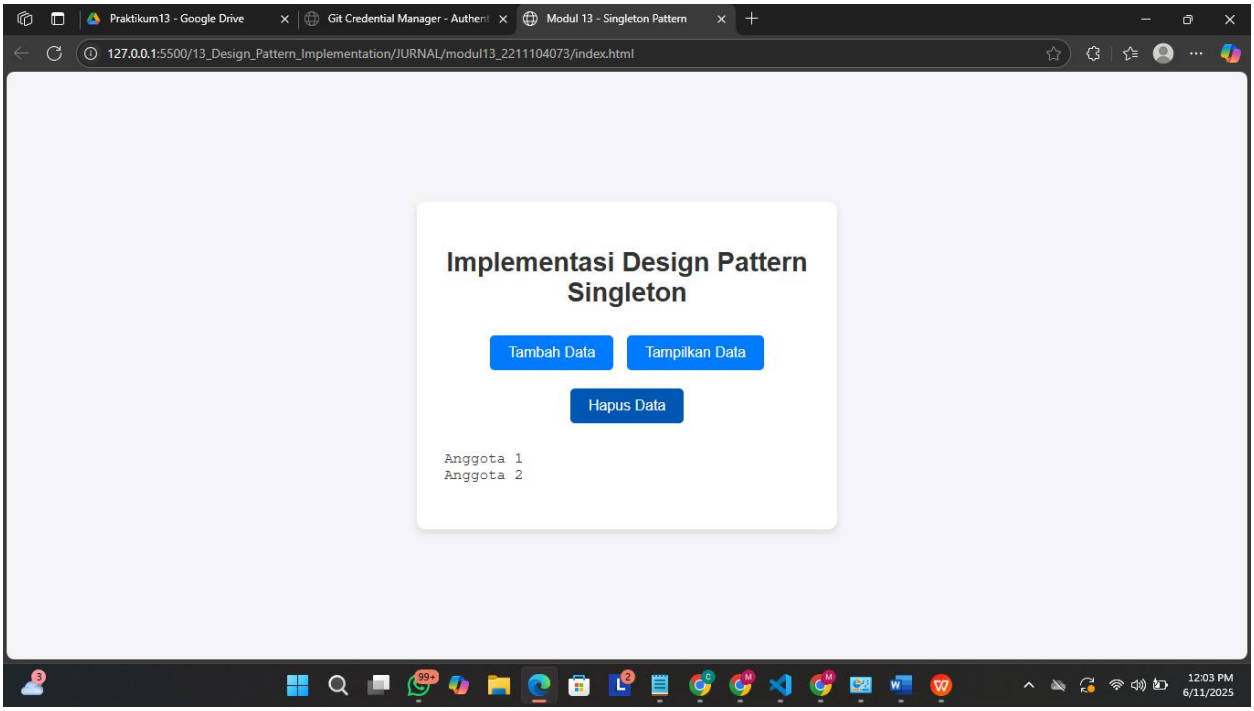
Penjelasan Keseluruhan

- **Kelas PusatDataSingleton:**
  - **Atribut DataTersimpan** adalah array yang digunakan untuk menyimpan data.
  - **Metode GetDataSingleton** mengembalikan instance tunggal dari kelas PusatDataSingleton.
  - **Metode AddSebuahData(input)** menambahkan data ke dalam list DataTersimpan.
  - **Metode HapusSebuahData(index)** menghapus data berdasarkan index.
  - **Metode PrintSemuaData()** mencetak semua data yang ada dalam DataTersimpan.
- **Program Utama:**
  - **Fungsi tambahData()** menambahkan data anggota kelompok dan asisten praktikum ke dalam list menggunakan instance Singleton.
  - **Fungsi tampilkanData()** menampilkan semua data yang ada di dalam list.
  - **Fungsi hapusData()** menghapus data berdasarkan index, dalam hal ini menghapus "Asisten Praktikum".
- **Urutan Pemanggilan:**
  - Pertama, tambahData() dipanggil untuk menambahkan data.
  - Kemudian, tampilkanData() dipanggil untuk menampilkan data yang telah ditambahkan.

- Selanjutnya, hapusData() dipanggil untuk menghapus data asisten praktikum.
- Terakhir, tampilkanData() dipanggil lagi untuk memastikan data asisten praktikum sudah terhapus.

Output




Praktikum13 - Google Drive
Git Credential Manager - Authent
Modul 13 - Singleton Pattern
127.0.0.1:5500/13_Design_Pattern_Implementation/JURNAL/modul13_2211104073/index.html
Implementasi Design Pattern Singleton
Tambah Data
Tampilkan Data
Hapus Data
Anggota 1
Anggota 2
12:03 PM
6/11/2025