Nama : Salman Alfarisi

Kelas : S1SE-07-01

NIM : 2311104036

B.A Dua Contoh Penggunaan Singleton

1. Logger (Pencatatan Log Aplikasi)
   Untuk mencatat log dari berbagai bagian aplikasi ke satu tempat, agar semua modul mengakses instance logger yang sama tanpa membuat duplikat.
2. Database Connection Pool
   Menghindari pembuatan koneksi baru setiap kali akses ke database. Singleton memastikan satu koneksi dipakai ulang dan lebih efisien.

B.B Langkah-Langkah Implementasi Singleton

1. Buat constructor yang bersifat private atau dikunci secara logika, sehingga tidak bisa diakses dari luar class secara bebas.
2. Buat properti static bernama _instance, yang akan menyimpan instance tunggal dari class tersebut.
3. Buat method static (misalnya getInstance() atau GetDataSingleton()) yang akan:
   - Mengecek apakah _instance sudah dibuat
   - Jika belum, buat instance baru
   - Kembalikan instance yang sudah ada
4. Gunakan instance tersebut di seluruh bagian aplikasi, tanpa membuat objek baru.

B.C Tiga Kelebihan dan Kekurangan Singleton

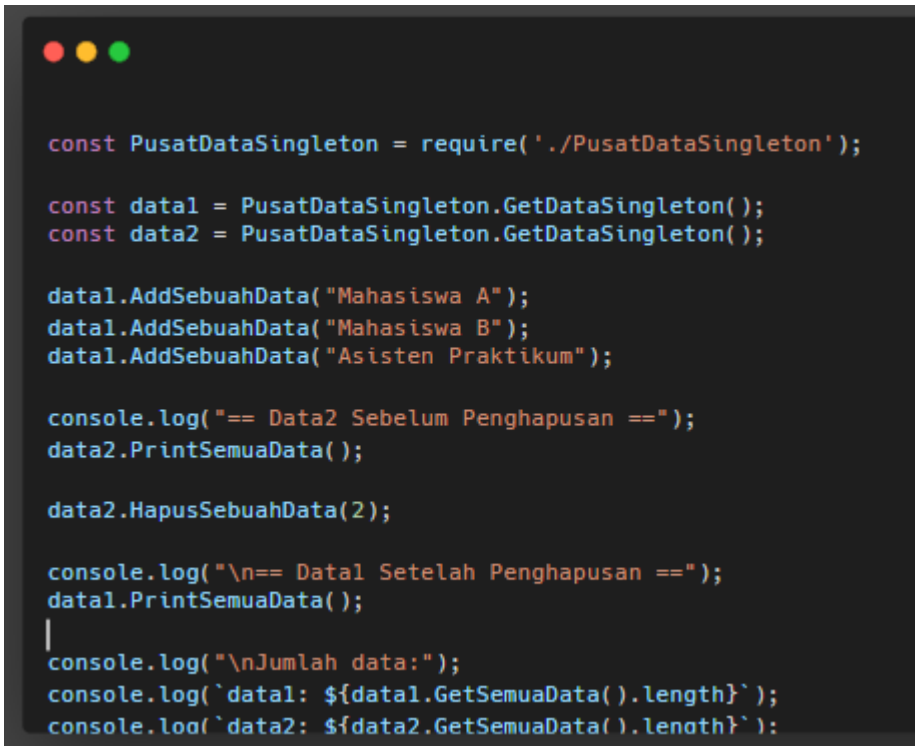
Kelebihan:

1. Menghemat resource: hanya satu instance yang dibuat.
2. Kontrol global: semua bagian program memakai instance yang sama.
3. Mudah diakses: instance dapat diakses melalui method static.

Kekurangan:

1. Menyulitkan pengujian unit test, karena sulit dibuat ulang dalam state berbeda.
2. Hidden dependency: class lain bisa bergantung pada instance global secara implisit.
3. Tidak cocok untuk multi-threading tanpa perlindungan khusus.

```
// file: PusatDataSingleton.js
class PusatDataSingleton {
  constructor() {
    if (PusatDataSingleton._instance) {
      return PusatDataSingleton._instance;
    }
    this.DataTersimpan = [];
    PusatDataSingleton._instance = this;
  }

  static GetDataSingleton() {
    if (!PusatDataSingleton._instance) {
      new PusatDataSingleton();
    }
    return PusatDataSingleton._instance;
  }

  GetSemuaData() {
    return this.DataTersimpan;
  }

  PrintSemuaData() {
    console.log("Data Tersimpan:");
    this.DataTersimpan.forEach((item, index) => {
      console.log(`${index + 1}. ${item}`);
    });
  }

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

  HapusSebuahData(index) {
    if (index >= 0 && index < this.DataTersimpan.length) {
      this.DataTersimpan.splice(index, 1);
    } else {
      console.log("Index tidak valid");
    }
  }
}

module.exports = PusatDataSingleton;
```



```
const PusatDataSingleton = require('./PusatDataSingleton');

const data1 = PusatDataSingleton.GetDataSingleton();
const data2 = PusatDataSingleton.GetDataSingleton();

data1.AddSebuahData("Mahasiswa A");
data1.AddSebuahData("Mahasiswa B");
data1.AddSebuahData("Asisten Praktikum");

console.log("== Data2 Sebelum Penghapusan ==");
data2.PrintSemuaData();

data2.HapusSebuahData(2);

console.log("\n== Data1 Setelah Penghapusan ==");
data1.PrintSemuaData();

console.log("\nJumlah data:");
console.log(`data1: ${data1.GetSemuaData().length}`);
console.log(`data2: ${data2.GetSemuaData().length}`);
```

Kode tersebut merupakan contoh penerapan pola desain Singleton dalam JavaScript menggunakan kelas PusatDataSingleton. Pola Singleton memastikan bahwa hanya ada satu instance dari suatu kelas yang dapat dibuat, dan semua bagian program yang membutuhkan objek tersebut akan mengakses instance yang sama. Dalam kode ini, dua variabel data1 dan data2 dideklarasikan dengan memanggil method GetDataSingleton(). Karena menggunakan pola Singleton, kedua variabel tersebut sebenarnya mengacu pada instance yang sama dari kelas PusatDataSingleton.

Selanjutnya, data ditambahkan melalui data1 dengan tiga entri: "Mahasiswa A", "Mahasiswa B", dan "Asisten Praktikum". Setelah itu, isi data dicetak menggunakan data2, dan hasilnya menunjukkan ketiga data tersebut. Kemudian data2 digunakan untuk menghapus elemen ketiga (indeks 2) yaitu "Asisten Praktikum". Ketika data dicetak ulang menggunakan data1, hasilnya menunjukkan bahwa perubahan yang dilakukan oleh data2 juga berdampak pada data1. Hal ini terjadi karena kedua variabel tersebut berbagi instance dan data yang sama. Terakhir, jumlah data yang ditampilkan melalui data1 dan data2 adalah 2, memperkuat bukti bahwa hanya ada satu sumber data yang digunakan secara bersama oleh kedua variabel tersebut sebuah karakteristik khas dari pola Singleton.