Novita Syahwa Tri Hapsari

2311104007

A. Berikan salah dua contoh kondisi dimana design pattern “Singleton” dapat digunakan.
    1. Saat aplikasi memerlukan hanya satu koneksi (atau satu pool) ke database agar penggunaan sumber daya optimal, maka Singleton memastikan hanya satu instance connection manager dibuat dan digunakan secara global.
    2. Sistem logging (pencatatan log)
    3. Agar semua bagian aplikasi mencatat ke satu tempat (file, console, atau remote logger), Singleton memberikan titik akses global ke logger dengan instance tunggal

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

    Jawab :

    1. Tambahkan private static field untuk menyimpan instance tunggal.
    2. Buat private constructor agar objek tidak bisa dibuat langsung via new.
    3. Buat public static method (biasanya getInstance()) untuk:
       Mengecek apakah instance sudah dibuat. Jika belum, buat instance dan simpan ke field statik. Kembalikan instance tersebut.
    4. Tambahkan mekanisme lazy initialization untuk menunda pembuatan objek hingga diperlukan. (opsional)
    5. Di lingkungan multithread tambahkan mekanisme sinkronisasi seperti lock atau double-checked locking agar instance hanya dibuat sekali meskipun dipanggil secara bersamaan

C. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.
    Jawab: Kelebihan: mudah dipanggil di mana saja di aplikasi tanpa perlu passing parameter, memastikan hanya ada satu objek dari kelas tersebut. Kekurangan: kode lain jadi tergantung langsung ke singleton, sulit di-ganti atau diuji, butuh penanganan threading

Code:

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

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

GetSemuaData() {
    return this.DataTersimpan;
}

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

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.");
    }
}
}

// ========== PROGRAM UTAMA ==========

// A & B
const data1 = PusatDataSingleton.GetDataSingleton();
const data2 = PusatDataSingleton.GetDataSingleton();

data1.AddSebuahData("Novita Syahwa");
data1.AddSebuahData("Rizky Firmansyah");
data1.AddSebuahData("Dian Prasetyo");
data1.AddSebuahData("Asisten Praktikum: Kak Revan");

console.log("\n--- Cetak Data melalui data2 ---");
data2.PrintSemuaData();
```

```
    const indexAsisten = data2.GetSemuaData().findIndex(data => data.includes("Asisten
Praktikum"));
    data2.HapusSebuahData(indexAsisten);

    console.log("\n--- Cetak Data melalui data1 setelah penghapusan ---");
    data1.PrintSemuaData();

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

Hasil running;

```
--- Cetak Data melalui data2 ---
Isi DataTersimpan:
1. Novita Syahwa
2. Rizky Firmansyah
3. Dian Prasetyo
4. Asisten Praktikum: Kak Revan

--- Cetak Data melalui data1 setelah penghapusan ---
Isi DataTersimpan:
1. Novita Syahwa
2. Rizky Firmansyah
3. Dian Prasetyo

Jumlah data di data1: 3
Jumlah data di data2: 3
```

Penjelasan dari code :

Kode tersebut membuat sebuah kelas bernama PusatDataSingleton yang menggunakan design pattern Singleton, artinya hanya akan dibuat satu objek saja meskipun dipanggil berkali-kali. Di dalam kelas ini terdapat sebuah list bernama DataTersimpan yang menyimpan data berupa nama-nama anggota atau informasi lainnya. Method GetDataSingleton() digunakan untuk mendapatkan objek tunggal dari kelas ini. Lalu, method AddSebuahData() digunakan untuk menambahkan data ke list, PrintSemuaData() untuk mencetak semua isi list ke layar, HapusSebuahData(index) untuk menghapus data berdasarkan urutan, dan GetSemuaData() untuk mengambil seluruh data dalam list.

Di bagian program utama, dua variabel (data1 dan data2) dibuat, namun keduanya merujuk pada objek yang sama karena menggunakan Singleton. Kemudian, data1 digunakan untuk menambahkan beberapa nama ke dalam list. Setelah itu, data2 mencetak semua data yang sudah ditambahkan. Nama asisten praktikum dihapus melalui data2, dan hasilnya dicek lagi lewat data1 untuk membuktikan bahwa perubahan di satu tempat memengaruhi keduanya (karena mereka satu objek yang sama). Terakhir, jumlah data ditampilkan untuk memastikan isi list sudah sesuai.