Tugas Jurnal Modul 13

1. Penjelasan Design Pattern Singleton
   A. Dua Kondisi Dimana design pattern singleton digunakan

      Logger (Pencatat Log Aplikasi)
      Sistem pencatatan log (logger) biasanya hanya membutuhkan satu instance saja agar semua bagian aplikasi mencatat ke tempat yang sama secara konsisten. Ini mencegah duplikasi file log dan menjaga konsistensi output log.

      Database Connection
      Saat mengakses database, biasanya hanya satu koneksi aktif yang dikelola dan dibagikan ke seluruh aplikasi. Menggunakan Singleton memastikan hanya satu objek koneksi database yang digunakan, mengurangi beban sistem dan mencegah konflik akses data.

   B. Langkah-langkah implementasi design pattern singleton

      Deklarasikan konstruktor sebagai private atau batasi akses langsungnya (di JavaScript, hal ini dicapai dengan menggunakan pengecekan instance secara manual karena tidak ada private constructor).

      - Buat atribut statis untuk menyimpan satu-satunya instance dari kelas tersebut.

      - Buat method statis (GetInstance) untuk mengakses instance tersebut. Jika instance belum ada, maka method ini akan membuatnya terlebih dahulu.

      - Seluruh akses ke kelas harus dilakukan melalui method ini, bukan dengan new.

   C. Kelebihan dan Kekurangan Singleton
      Kelebihan:
      - Kontrol Global Instance: Menjamin hanya satu objek yang dibuat dan digunakan bersama di seluruh aplikasi.

      - Efisiensi Sumber Daya: Cocok untuk resource mahal seperti koneksi database atau konfigurasi global.

      - Kemudahan Akses: Mudah diakses dari bagian mana pun dalam program tanpa harus menyuntikkan objek ke setiap kelas.

      Kekurangan:
      - Menyulitkan Testing (Unit Test): Karena global state, sulit untuk mengganti instance selama testing.

      - Pelanggaran Prinsip Single Responsibility: Karena kelas bertanggung jawab atas dirinya sendiri dan juga manajemen instance-nya.

- Menyembunyikan Dependensi: Karena bisa diakses langsung, dependensi antar komponen jadi tidak terlihat jelas dan sulit untuk diatur.

2. DataSingleton.js

```
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 Data:");
    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.");
    }
  }
}

module.exports = PusatDataSingleton;
```

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.

3. Main.js

```
const PusatDataSingleton = require("./DataSingleton");

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

data1.AddSebuahData("Nama Anggota 1");
data1.AddSebuahData("Nama Anggota 2");
data1.AddSebuahData("Nama Asisten Praktikum");

console.log("\nCetak dari data2:");
data2.PrintSemuaData();

data2.HapusSebuahData(2);

console.log("\nSetelah penghapusan (dari data1):");
data1.PrintSemuaData();

console.log("\nJumlah data:");
console.log("Data1 count:", data1.GetSemuaData().length);
console.log("Data2 count:", data2.GetSemuaData().length);
```

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.