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.

2. 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.

B. 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`.

C. 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

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.