1. Singleton Design Pattern

Singleton Pattern, memastikan bahwa sebuah kelas hanya dapat memiliki satu objek sepanjang jalannya program. Pola ini juga menyediakan satu titik akses global untuk objek tersebut.

Singleton Design Pattern biasanya digunakan pada Logger pada aplikasi, karena biasanya logger hanya membutuhkan satu instance agar semua modul aplikasi mencatat log ke tempat yang sama secara konsisten, hal ini mencegah file yang terduplikat.

Selain itu pattern ini juga bisa digunakan pada Database connection untuk mengakses database. Nantinya akan terdapat satu koneksi aktif yang nantinya aksesnya akan disebar.

Kelebihan Singleton :

1. Kontrol yang lebih tersentralisir, karena menjamin hanya satu objek yang dibuat dan digunakan Bersama di seluruh aplikasi.
2. Efisien Sumber Daya.
3. Mudah Diakses

Kekurangan :

1. Menyulitkan unit testing. Unit testing akan menjadi rumit karena akan dengan global state akan sulit mengganti instance pada testing.
2. Kemungkinan terjadi pelanggaran prinsip single responsibility.
3. Menyembunyikan dependensi.

DataSingleton.js

Jurnal > JS 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 diatas adalah implementasi dari pola singleton yang memastikan hanya ada satu object yang digunakan dari seluruh aplikasi.

Instance nantinya akan disimpan pada object statis. Class ini nantinya akan menyimpan data pada array serta terdapat method untuk CRUD data dan show data, dengan begitu perubahan akan bersifat global.

Main.js

```
Jurnal > JS main.js > ...
  1  const PusatDataSingleton = require("./DataSingleton");
  2
  3  const data1 = PusatDataSingleton.GetDataSingleton();
  4  const data2 = PusatDataSingleton.GetDataSingleton();
  5
  6  data1.AddSebuahData("Nama Anggota 1");
  7  data1.AddSebuahData("Nama Anggota 2");
  8  data1.AddSebuahData("Nama Asisten Praktikum");
  9
 10  console.log("\nCetak dari data2:");
 11  data2.PrintSemuaData();
 12
 13  data2.HapusSebuahData(2);
 14
 15  console.log("\nSetelah penghapusan (dari data1):");
 16  data1.PrintSemuaData();
 17
 18  console.log("\nJumlah data:");
 19  console.log("Data1 count:", data1.GetSemuaData().length);
 20  console.log("Data2 count:", data2.GetSemuaData().length);
```

Penjelasan :

Pada kode diatas terdapat fungsi pemanggilan dari kelas PusatDataSingleton, yang di dalamnya akan dipanggil data1 dan data2. Kemudian data tersebut akan dideklarasikan dan diacukan pada satu instance yang sama. Kemudian dari data1 akan ditambahkan 3 data yang isinya dikeluarkan menggunakan data2, dan dihapus melalui data2.
Kemudian data dicetak Kembali menggunakan data1, kemudian keduanya akan di print masing2 yakni cetak data1 dan cetak data2.

Output :

```
01\13_Design_Pattern\Jurnal> node main.js

Cetak dari data2:
Isi Data:
1. Nama Anggota 1
2. Nama Anggota 2
3. Nama Asisten Praktikum

Setelah penghapusan (dari data1):
Isi Data:
1. Nama Anggota 1
2. Nama Anggota 2

Jumlah data:
Data1 count: 2
Data2 count: 2
PS D:\Campuss\SMT 4\Konstruksi Perangkat Lu
```