JURNAL MODUL 13

2311104008
Viona Aziz Syahputri

Link Github
https://github.com/viona123/KPL_Viona-Aziz-Syahputri_2311104008_SE07-01/tree/main/13_Design_Pattern_Implementation/Jurnal_GUI_2311104008

main.js

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

// Membuat dua variabel singleton
const data1 = PusatDataSingleton.getDataSingleton();
const data2 = PusatDataSingleton.getDataSingleton();

// Menambahkan data ke data1
data1.addSebuahData("Alice (Anggota)");
data1.addSebuahData("Bob (Anggota)");
data1.addSebuahData("Charlie (Asisten Praktikum)");

// Menampilkan data dari data2
console.log("== Data dari data2 ==");
data2.printSemuaData();

// Menghapus asisten praktikum dari data2
data2.hapusSebuahData(2); // index 2 untuk "Charlie"

// Menampilkan data dari data1 setelah penghapusan
console.log("\n== Data dari data1 setelah penghapusan ==");
data1.printSemuaData();

// Menampilkan jumlah data
console.log("\nJumlah data di data1:", data1.getSemuaData().length);
console.log("Jumlah data di data2:", data2.getSemuaData().length);

```

Dua variabel, data1 dan data2, sama-sama mengambil objek dari PusatDataSingleton, tapi sebenarnya keduanya menunjuk ke objek yang sama. Saat menambahkan data melalui data1, datanya juga otomatis bisa dilihat lewat data2. Setelah itu, data "Charlie" dihapus lewat data2, dan hasilnya juga langsung berubah saat dicek lewat data1, karena memang mereka mengakses tempat penyimpanan yang sama. Singkatnya, ini contoh gimana sebuah data bisa dipakai dan dimodifikasi bareng dari beberapa tempat, tapi tetap konsisten karena cuma ada satu sumber.

PusatDataSingleton.js

```
class PusatDataSingleton {
  constructor() {
    if (PusatDataSingleton._instance) {
      throw new Error("Gunakan PusatDataSingleton.getDataSingleton()");
    }
    this.DataTersimpan = [];
  }

  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;
```

Di kelas PusatDataSingleton, objek hanya bisa diambil lewat method getDataSingleton() dan tidak bisa dibuat langsung menggunakan new karena akan memunculkan error. Kelas ini menyimpan data dalam array DataTersimpan, serta menyediakan beberapa method untuk menambah data (addSebuahData()), menghapus data berdasarkan index (hapusSebuahData()), mencetak semua isi data (printSemuaData()), dan mengambil seluruh data (getSemuaData()). Karena hanya ada satu instance yang dipakai bersama, semua perubahan data akan terlihat sama dari manapun objek ini diakses. Jadi intinya, ini seperti punya satu pusat penyimpanan data yang bisa dipakai bareng oleh seluruh bagian program tanpa bikin salinan baru.

Output

```
== Data dari data2 ==
Isi Data:
1. Alice (Anggota)
2. Bob (Anggota)
3. Charlie (Asisten Praktikum)

== Data dari data1 setelah penghapusan ==
Isi Data:
1. Alice (Anggota)
2. Bob (Anggota)

Jumlah data di data1: 2
Jumlah data di data2: 2
```