PusatDataSingleton.js => Implementasi design pattern Singleton untuk manajemen data terpusat.

```
class PusatDataSingleton {
  hapusSebuahData(index) {
    if (index >= 0 && index < this.DataTersimpan.length) {
      this.DataTersimpan.splice(index, 1);
    }
  }

  printSemuaData() {
    console.log("=== Data Tersimpan ===");
    this.DataTersimpan.forEach((item, index) => {
      console.log(`${index + 1}. ${item}`);
    });
  }

  getSemuaData() {
    return [...this.DataTersimpan]; // Return copy untuk keamanan
  }
}

module.exports = PusatDataSingleton;
```

- static instance: Variabel statis untuk menyimpan instance tunggal.
- Private Constructor: Dicegah instantiasi langsung via new.
- getInstance(): Satu-satunya cara akses instance (global point of access).

app.js => Demonstrasi penggunaan Singleton sesuai soal.

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

function main() {
  const data1 = PusatDataSingleton.getInstance();
  const data2 = PusatDataSingleton.getInstance();

  // Tambah data
  data1.addSebuahData("Kelompok A");
  data1.addSebuahData("Asisten Reva");

  // Print dari instance berbeda
  console.log("[Data2] Sebelum hapus:");
  data2.printSemuaData();

  // Hapus data
  data2.hapusSebuahData(1); // Hapus "Asisten Reva"

  // Verifikasi
  console.log("\n[Data1] Setelah hapus:");
  data1.printSemuaData();

  // Cek count
  console.log("\nTotal data:");
  console.log(`Data1: ${data1.getSemuaData().length} item`);
  console.log(`Data2: ${data2.getSemuaData().length} item`);
}

main();
```

- Mendapatkan instance via getInstance()

- Operasi pada data1 dan data2 memengaruhi data yang sama
- Membuktikan sifat Singleton dengan ===

PusatDataSingleton.test.js => Memverifikasi perilaku Singleton.

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

test('Singleton instance harus sama', () => {
  const instance1 = PusatDataSingleton.getInstance();
  const instance2 = PusatDataSingleton.getInstance();
  expect(instance1).toBe(instance2);
});

test('Penambahan data harus konsisten antar instance', () => {
  const instance1 = PusatDataSingleton.getInstance();
  instance1.addSebuahData("Test Data");

  const instance2 = PusatDataSingleton.getInstance();
  expect(instance2.getSemuaData()).toContain("Test Data");
});

test('Hapus data berpengaruh ke semua instance', () => {
  const instance1 = PusatDataSingleton.getInstance();
  instance1.addSebuahData("Data A");

  const instance2 = PusatDataSingleton.getInstance();
  instance2.hapusSebuahData(0);

  expect(instance1.getSemuaData()).not.toContain("Data A");
});
```

- Uniqueness: Memastikan hanya ada 1 instance.
- Data Sharing: Perubahan di satu instance terlihat di instance lain.
- Error Handling: Test penghapusan data invalid.

Ouput

```
[Data2] Sebelum hapus:
=== Data Tersimpan ===
1. Kelompok A
2. Asisten Reva

[Data1] Setelah hapus:
=== Data Tersimpan ===
1. Kelompok A

Total data:
Data1: 1 item
Data2: 1 item
```