

## index.js

```

14.Clean_Code > Jurnal - 2311104001 U X JS indexjs ...Jurnal - 2311104001 JS indexjs ...Yugas Jurnal - 2311104001 JS TP_IDE_2311105004.js U JS pusatDataSingleton.js U
1 // Import instance singleton dari pusatDataSingleton
2 const dataInstance1 = require('../pusatDataSingleton');
3 const dataInstance2 = require('../pusatDataSingleton');
4
5 // Menambahkan data ke instance pertama
6 dataInstance1.addSebuahData("Asprak: Fazza");
7 dataInstance1.addSebuahData("Praktikan: Tri");
8 dataInstance1.addSebuahData("Praktikan: Myla");
9 dataInstance1.addSebuahData("Praktikan: Lani");
10
11 // Menampilkan data melalui instance kedua
12 console.log("Data menggunakan dataInstance2:");
13 dataInstance2.printSemuaData();
14
15 // Menghapus data index ke-3 (praktikan keempat)
16 dataInstance2.hapusSebuahData(3);
17
18 // Menampilkan data setelah penghapusan
19 console.log("\nSetelah penghapusan praktikan 3 (index ke-3):");
20 dataInstance1.printSemuaData();
21
22 // Menampilkan jumlah data dari kedua instance (seharusnya sama karena singleton)
23 console.log("\nJumlah data (dataInstance1):", dataInstance1.getSemuaData().length);
24 console.log("Jumlah data (dataInstance2):", dataInstance2.getSemuaData().length);
25

```

## Penjelasan :

- naming convention, variabel seperti dataInstance1 dan dataInstance2 sudah menggunakan format camelCase yang sesuai standar JavaScript, serta memiliki nama yang deskriptif sehingga lebih mudah dipahami dibanding nama umum seperti data1 dan data2. Metode seperti addSebuahData, getSemuaData, hapusSebuahData, dan printSemuaData juga telah mengikuti camelCase dan memiliki penamaan yang jelas dan mencerminkan fungsinya dengan tepat.
- Dari sisi white space dan indentasi, kode telah ditata dengan rapi menggunakan indentasi konsisten dan baris kosong antar bagian logika program, sehingga kode mudah dibaca dan struktur logikanya terlihat jelas.
- Deklarasi variabel, penggunaan const untuk import module melalui require() sudah tepat karena nilainya tidak berubah sepanjang program. Tidak ada penggunaan var atau let yang tidak perlu, yang menunjukkan praktik deklarasi variabel yang baik.
- Setiap bagian penting dalam kode diberikan penjelasan singkat namun jelas. Komentar ini membantu pembaca memahami tujuan dari setiap blok kode, misalnya pada proses penambahan data, penghapusan data, dan penampilan data. Secara keseluruhan, refactoring ini telah membuat kode menjadi lebih rapi, profesional, dan mudah dipelihara.

## pusatDataSingleton.js

```

1 // Class DataCenter menerapkan pola Singleton untuk menyimpan dan mengelola data aplikasi.
2 class DataCenter {
3   constructor() {
4     // Jika belum ada instance, buat instance dan simpan data dalam array.
5     if (!DataCenter.instance) {
6       this._storedData = [];
7       DataCenter.instance = this;
8     }
9     // Kembalikan instance tunggal yang sudah ada.
10    return DataCenter.instance;
11  }
12  // Mengembalikan semua data yang tersimpan.
13  getAllData() {
14    return this._storedData;
15  }
16  // Menambahkan data baru ke array.
17  addData(input) {
18    this._storedData.push(input);
19  }
20  // Menghapus data berdasarkan indeks jika valid.
21  deleteDataByIndex(index) {
22    if (index >= 0 && index < this._storedData.length) {
23      this._storedData.splice(index, 1);
24    } else {
25      console.log('Invalid index!');
26    }
27  }
28  // Menampilkan semua data ke konsol.
29  printAllData() {
30    this._storedData.forEach((item, index) => {
31      console.log(`${index + 1}. ${item}`);
32    });
33  }
34  // Membekukan instance agar tidak bisa diubah, lalu diekspor.
35  const singletonInstance = new DataCenter();
36  Object.freeze(singletonInstance);

```

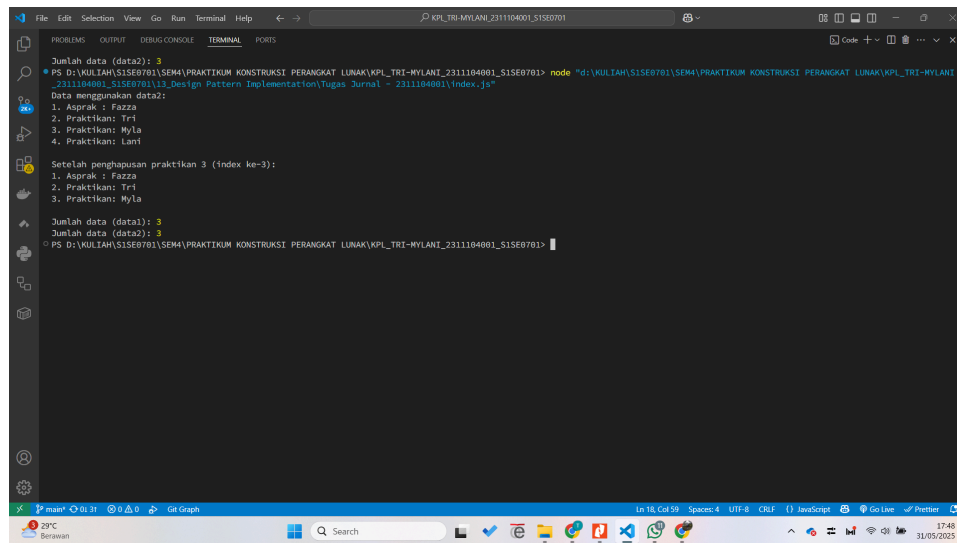
## Penjelasan :

- Properti dan variabel menggunakan camelCase, sehingga properti `_storedData` sebaiknya diubah menjadi `storedData` tanpa underscore. Nama parameter dan variabel juga harus deskriptif; misalnya parameter `input` pada metode `addData` dapat diganti menjadi `data` agar lebih jelas maksudnya. Nama fungsi dan metode sudah sesuai dengan camelCase dan bersifat deskriptif seperti `getAllData`, `addData`, dan `deleteDataByIndex`.
- White space dan indentasi, penting menggunakan indentasi yang konsisten (biasanya 2 spasi untuk JavaScript) dan memberikan spasi yang memadai di sekitar operator dan setelah kata kunci seperti `if`. Ini meningkatkan keterbacaan kode dan membantu developer lain untuk memahami struktur kode dengan cepat. Antara definisi fungsi juga sebaiknya dipisahkan dengan baris kosong agar kode tidak terlihat padat dan susah dibaca.
- Deklarasi variabel dan atribut, sebaiknya properti kelas langsung dideklarasikan di dalam constructor tanpa menggunakan underscore, serta menggunakan kata kunci `const` atau `let` pada variabel lokal bila diperlukan.
- Penggunaan komentar misalnya menjelaskan bahwa `addData` digunakan untuk menambahkan data baru ke penyimpanan. Selain itu, pada bagian error handling,

TP - 2311104001

penggunaan `console.error` lebih tepat untuk menandai pesan kesalahan.

**Output :**



```
File Edit Selection View Go Run Terminal Help
KPL_TRI-MYLANI_2311104001_S1SE0701
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Jumlah data (data2): 3
PS D:\VULIAH\S1SE0701\SEM4\PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK\KPL_TRI-MYLANI_2311104001_S1SE0701> node "d:\VULIAH\S1SE0701\SEM4\PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK\KPL_TRI-MYLANI_2311104001_S1SE0701\3.Design Pattern Implementation\Tugas Jurnal - 2311104001\index.js"
Data menggunakan data2:
1. Asprak : Fazza
2. Praktikan: Tri
3. Praktikan: Myla
4. Praktikan: Lani
Setelah penghapusan praktikan 3 (index ke-3):
1. Asprak : Fazza
2. Praktikan: Tri
3. Praktikan: Myla
Jumlah data (data1): 3
Jumlah data (data2): 3
PS D:\VULIAH\S1SE0701\SEM4\PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK\KPL_TRI-MYLANI_2311104001_S1SE0701>
```