

1. Penjelasan Design Pattern

A. Berikan salah dua contoh kondisi dimana design pattern “Singleton” dapat digunakan.

Logger dan saat pengaturan koneksi database.

B. Berikan penjelasan singkat mengenai langkah-langkah dalam mengimplementasikan design pattern “Singleton”.

- Membuat kelas dengan konstruktor private
- Menyediakan variabel statis untuk menyimpan instance yang Tunggal
- Menyediakan method statis

C. Berikan tiga kelebihan dan kekurangan dari design pattern “Singleton”.

- Kelebihan
 - Kontrol akses ke instance Tunggal
 - Mudah diakses secara global
 - Menghemat sumber daya karena tidak membuat banyak objek yang hampir sama
- Kekurangan
 - Menyulitkan unit testing
 - Menyembunyikan dependensi
 - Masalah pada multithreading

2. Penjelasan Singleton

Kode ini mengimplementasikan design pattern Singleton dalam JavaScript dengan kelas `PusatDataSingleton` yang hanya memungkinkan satu instance dibuat dan diakses melalui method statis `GetDataSingleton()`. Kelas ini menyimpan data dalam list `DataTersimpan` dan menyediakan method untuk menambah, menghapus, mencetak, serta mengambil seluruh data. Dalam program utama, dua variabel `data1` dan `data2` mengakses instance yang sama, sehingga perubahan data lewat salah satu variabel langsung terlihat pada yang lain, membuktikan pola Singleton berjalan dengan benar.

```
JS PusatDataSingleton.js U X
13_Design_Pattern > JURNAL_2311104013 > JS PusatDataSingleton.js > ...
1  class PusatDataSingleton {
2      constructor() {
3          if (PusatDataSingleton._instance) {
4              return PusatDataSingleton._instance;
5          }
6          this.DataTersimpan = [];
7          PusatDataSingleton._instance = this;
8      }
9
10     static GetDataSingleton() {
11         if (!PusatDataSingleton._instance) {
12             PusatDataSingleton._instance = new PusatDataSingleton();
13         }
14         return PusatDataSingleton._instance;
15     }
16
17     GetSemuaData() {
18         return this.DataTersimpan;
19     }
20
21     PrintSemuaData() {
22         if (this.DataTersimpan.length === 0) {
23             console.log("DataTersimpan kosong");
24             return;
25         }
26         this.DataTersimpan.forEach((item, index) => {
27             console.log(`${index}: ${item}`);
28         });
29     }
30
31     AddSebuahData(input) {
32         this.DataTersimpan.push(input);
33     }
34
35     HapusSebuahData(index) {
36         if (index >= 0 && index < this.DataTersimpan.length) {
37             this.DataTersimpan.splice(index, 1);
38         } else {
39             console.log(`Index ${index} tidak valid`);
40         }
41     }
42 }
43
44 const data1 = PusatDataSingleton.GetDataSingleton();
45 const data2 = PusatDataSingleton.GetDataSingleton();
46
47 data1.AddSebuahData("Anggota Kelompok 1");
48 data1.AddSebuahData("Anggota Kelompok 2");
49 data1.AddSebuahData("Asisten Praktikum");
50
51 console.log("Isi data 2 sebelum hapus:");
52 data2.PrintSemuaData();
53
54 data2.HapusSebuahData(2);
55
56 console.log(`\nIsi data 1 setelah hapus Asisten Praktikum:`);
57 data1.PrintSemuaData();
58
59 console.log(`\nJumlah data di data 1:`, data1.GetSemuaData().length);
60 console.log(`Jumlah data di data 2:`, data2.GetSemuaData().length);
61
```

3. Output

```
● PS E:\KULIAH\Semester 4\Konstruksi Perangkat Lunak\Praktik
  Isi data 2 sebelum hapus:
  0: Anggota Kelompok 1
  1: Anggota Kelompok 2
  2: Asisten Praktikum

  Isi data 1 setelah hapus Asisten Praktikum:
  0: Anggota Kelompok 1
  1: Anggota Kelompok 2

  Jumlah data di data 1: 2
  Jumlah data di data 2: 2
○ PS E:\KULIAH\Semester 4\Konstruksi Perangkat Lunak\Praktik
```