

Penjelasan Observer Design Pattern

1. Contoh Kondisi Penggunaan

Ketika sebuah aplikasi hanya membutuhkan satu koneksi database yang dibagi ke seluruh bagian sistem (untuk efisiensi dan konsistensi), maka Singleton cocok digunakan agar hanya ada satu instance koneksi database yang aktif.

2. Langkah-langkah Implementasi

- a. Buat field privat statis di dalam kelas untuk menyimpan satu-satunya instance Singleton.
- b. Buat konstruktor privat, agar tidak bisa dibuat instance dari luar kelas.
- c. Buat metode statis publik (misalnya `getInstance`) untuk mengakses instance Singleton.
- d. Di dalam metode `getInstance`, gunakan lazy initialization, yaitu membuat instance hanya ketika dibutuhkan pertama kali.
- e. Semua kode klien harus mengakses Singleton melalui `getInstance`, bukan membuat objek baru.

3. Kelebihan & Kekurangan

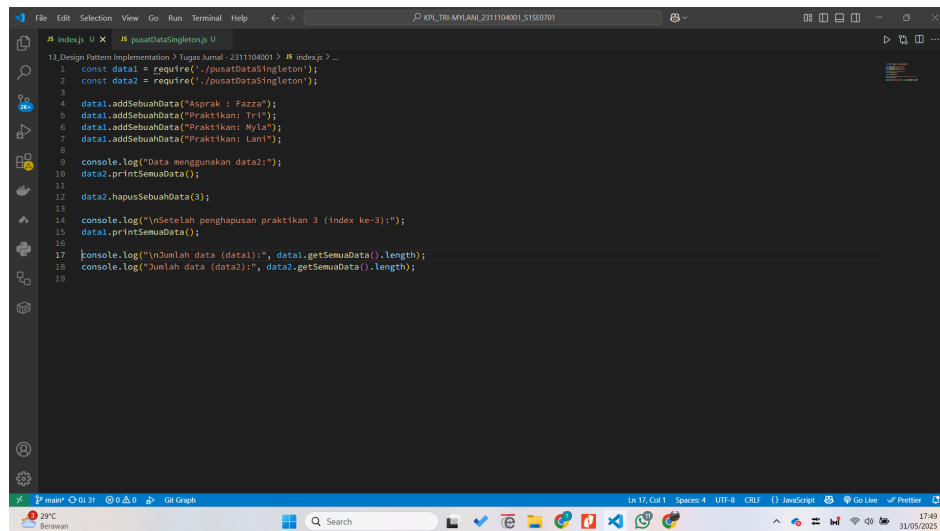
a. Kelebihan

- Kontrol penuh atas instance – Memastikan hanya ada satu objek yang digunakan bersama.
- Akses global – Instance dapat diakses dari mana pun dalam aplikasi.
- Efisiensi memori – Menghindari duplikasi objek yang seharusnya hanya ada satu (misalnya, logger, config, dll).

b. Kekurangan

- Melanggar Prinsip Tanggung Jawab Tunggal – Singleton mengelola logika dirinya sekaligus akses global.
- Sulit diuji (unit testing) – Karena konstruktor privat dan instance bersifat global, sulit dibuat tiruan (mock).
- Berpotensi menyebabkan masalah pada multithreading – Jika tidak diatur dengan benar, bisa terjadi dua objek singleton yang dibuat secara bersamaan.

index.js



```

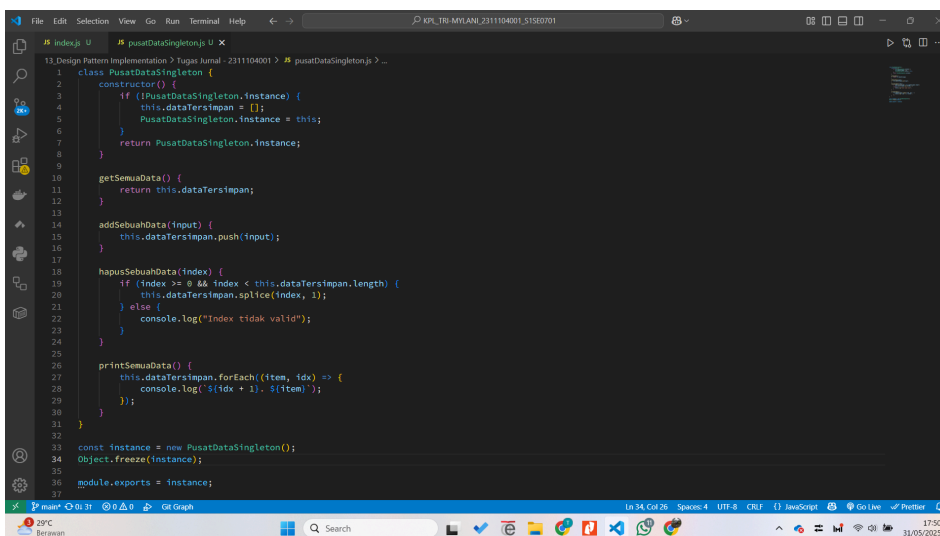
13_Design Pattern Implementation > Tugas Jurnal - 2311104001 > index.js > ...
1  const data1 = require('./pusatDataSingleton');
2  const data2 = require('./pusatDataSingleton');
3
4  data1.addSebuahData("Asprak : Fazza");
5  data1.addSebuahData("Praktikan: Tri");
6  data1.addSebuahData("Praktikan: Myta");
7  data1.addSebuahData("Praktikan: Lani");
8
9  console.log("Data menggunakan data2:");
10 data2.printSemuaData();
11
12 data2.hapusSebuahData(3);
13
14 console.log("\nSetelah penghapusan praktikan 3 (index ke-3):");
15 data1.printSemuaData();
16
17 console.log("\nJumlah data (data1):", data1.getSemuaData().length);
18 console.log("Jumlah data (data2):", data2.getSemuaData().length);
19

```

Penjelasan :

Kode tersebut merupakan implementasi penggunaan design pattern Singleton dalam konteks Node.js, di mana modul pusatDataSingleton hanya dibuat satu kali instance dan dibagikan ke semua pemanggilnya. Variabel data1 dan data2 sama-sama mengimpor instance Singleton yang sama dari file pusatDataSingleton. Pada data1, beberapa data ditambahkan, termasuk nama asisten dan praktikan. Kemudian, data2 digunakan untuk mencetak data yang sama (karena referensinya satu instance), lalu menghapus data pada indeks ke-3. Saat data1 mencetak ulang data, hasilnya sudah berubah, menunjukkan bahwa data1 dan data2 memang menunjuk ke objek Singleton yang sama. Di akhir, jumlah data dari data1 dan data2 juga sama, menegaskan bahwa perubahan dari satu variabel langsung tercermin pada yang lain.

pusatDataSingleton.js



```

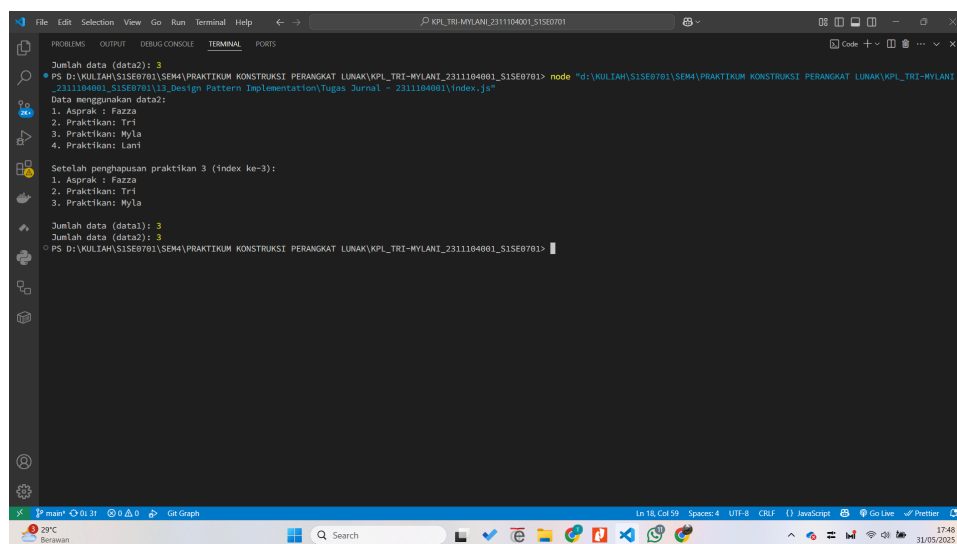
13_Design Pattern Implementation > Tugas Jurnal - 2311104001 > pusatDataSingleton.js > ...
1  class PusatDataSingleton {
2    constructor() {
3      if (!PusatDataSingleton.instance) {
4        this.dataTersimpan = [];
5        PusatDataSingleton.instance = this;
6      }
7      return PusatDataSingleton.instance;
8    }
9
10   getSemuaData() {
11     return this.dataTersimpan;
12   }
13
14   addSebuahData(input) {
15     this.dataTersimpan.push(input);
16   }
17
18   hapusSebuahData(index) {
19     if (index >= 0 && index < this.dataTersimpan.length) {
20       this.dataTersimpan.splice(index, 1);
21     } else {
22       console.log("Index tidak valid");
23     }
24   }
25
26   printSemuaData() {
27     this.dataTersimpan.forEach((item, idx) => {
28       console.log(`${idx + 1}. ${item}`);
29     });
30   }
31 }
32
33 const instance = new PusatDataSingleton();
34 Object.freeze(instance);
35
36 module.exports = instance;
37

```

Penjelasan :

Kode di atas adalah implementasi pola desain Singleton dalam JavaScript untuk kelas bernama PusatDataSingleton. Kelas ini memastikan hanya ada satu objek instance yang dibuat dan digunakan sepanjang aplikasi berjalan. Di dalamnya terdapat properti dataTersimpan sebagai tempat menyimpan data dalam bentuk array, serta beberapa metode untuk mengelola data tersebut, seperti menambahkan data (addSebuahData), menghapus data berdasarkan indeks (hapusSebuahData), mengambil semua data (getSemuaData), dan menampilkan semua data ke konsol (printSemuaData). Setelah instance dibuat, objek tersebut dibekukan dengan Object.freeze() agar tidak bisa diubah, menjaga konsistensi singleton. Terakhir, instance ini diekspor untuk digunakan di modul lain.

Output :



```
File Edit Selection View Go Run Terminal Help
KPL_TRI-MYLANI_2311104001_SISE0701

Jumlah data (data2): 3
PS D:\VULIAR\SISE0701\SEMA\PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK\KPL_TRI-MYLANI_2311104001_SISE0701> node "d:\VULIAR\SISE0701\SEMA\PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK\KPL_TRI-MYLANI_2311104001_SISE0701\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:\VULIAR\SISE0701\SEMA\PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK\KPL_TRI-MYLANI_2311104001_SISE0701>
```