

Nama : M. Faris

Nim : 2311104017

Jurnal Modul 13

A. Contoh kondisi di mana design pattern Singleton dapat digunakan

1. Koneksi Database
Saat aplikasi membutuhkan satu koneksi database yang dipakai bersama oleh banyak bagian program, agar tidak membuat koneksi berulang-ulang dan membuang resource.
2. Manajemen Konfigurasi Aplikasi
Bila aplikasi memiliki konfigurasi yang harus diakses secara global dan konsisten, seperti pengaturan sistem, maka Singleton memastikan hanya ada satu objek konfigurasi yang digunakan di seluruh aplikasi.

B. Langkah-langkah singkat dalam mengimplementasikan design pattern Singleton

1. Membuat constructor kelas menjadi private
2. Membuat atribut statis untuk menyimpan instance tunggal
3. Membuat method statis (misalnya getInstance)
4. Penggunaan instance melalui method statis tersebut

C. Tiga kelebihan dan kekurangan design pattern Singleton

Kelebihan:

1. Memastikan hanya ada satu instance objek
2. Menyediakan akses global ke instance tersebut
3. Inisialisasi objek dilakukan secara lazy (hanya saat diperlukan)

Kekurangan:

1. Melanggar prinsip Single Responsibility Principle (SRP)
2. Sulit diuji (unit test) karena constructor private dan instance global
3. Perlu perhatian khusus di lingkungan multithreaded

1. Source Kode

```
class PusatDataSingleton {
  constructor() {
    this.DataTersimpan = [];
  }

  static getDataSingleton() {
    if (!PusatDataSingleton._instance) {
      PusatDataSingleton._instance = new PusatDataSingleton();
    }
    return PusatDataSingleton._instance;
  }

  getSemuaData() {
    return this.DataTersimpan;
  }

  printSemuaData() {
    this.DataTersimpan.forEach((data) => console.log(data));
  }

  addSebuahData(input) {
    this.DataTersimpan.push(input);
  }

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

const data1 = PusatDataSingleton.getDataSingleton();
const data2 = PusatDataSingleton.getDataSingleton();

data1.addSebuahData("Anggota Kelompok 1");
data1.addSebuahData("Anggota Kelompok 2");
data1.addSebuahData("Asisten Praktikum");

data2.printSemuaData();
```

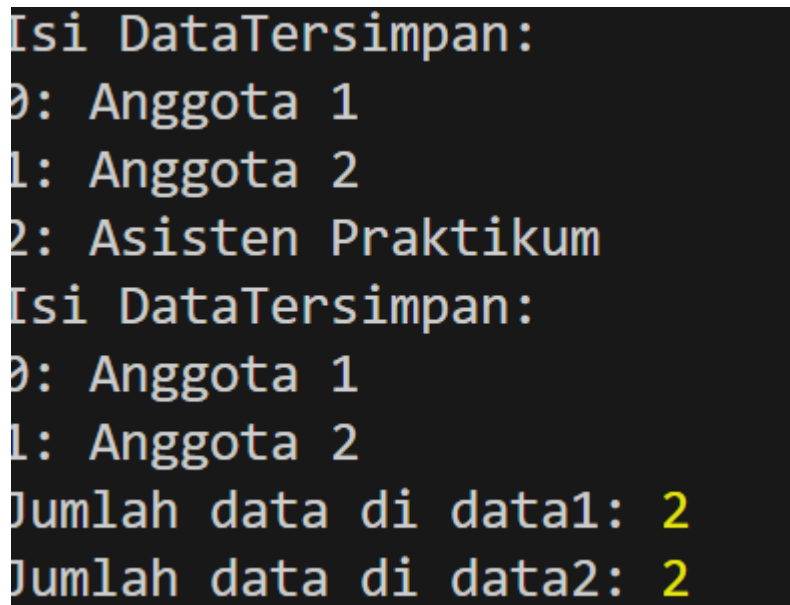
```
data2.hapusSebuahData(2);
```

```
data1.printSemuaData();
```

```
console.log("Jumlah data pada data1:", data1.getSemuaData().length);
```

```
console.log("Jumlah data pada data2:", data2.getSemuaData().length);
```

2. Output :



```
Isi DataTersimpan:
0: Anggota 1
1: Anggota 2
2: Asisten Praktikum
Isi DataTersimpan:
0: Anggota 1
1: Anggota 2
Jumlah data di data1: 2
Jumlah data di data2: 2
```

3. Penjelsan :

Class PusatDataSingleton menggunakan design pattern Singleton untuk memastikan hanya ada satu instance dari kelas tersebut selama program berjalan. Instance tunggal ini menyimpan daftar data berupa string pada atribut DataTersimpan, yang dapat diakses dan dimodifikasi melalui method seperti addSebuahData, hapusSebuahData, dan printSemuaData. Dengan memanggil getDataSingleton(), program akan selalu mendapatkan objek yang sama, sehingga perubahan yang dilakukan melalui satu variabel (misalnya data1) akan langsung terlihat pada variabel lain (data2).