Muhammad fathi rafa

2311104022

Jurnal Modul 14

Saya menggunakan tugas jurnal 13 agar menjadi standar code dari .NET

Pusat data singleton.js

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

  /**
   * Tambahkan sebuah data ke dalam array data
   * @param {string} item - Data yang akan ditambahkan
   */
  addData(item) {
    this._data.push(item);
  }

  /**
   * Hapus data berdasarkan indeks
   * @param {number} index - Indeks data yang ingin dihapus
   */
  removeData(index) {
    if (index >= 0 && index < this._data.length) {
      this._data.splice(index, 1);
    } else {
      console.warn('Index out of bounds saat menghapus data');
    }
  }

  /**
   * Cetak semua data yang ada ke console
   */
  printAllData() {
    this._data.forEach((item, idx) => {
      console.log(`[${idx}] ${item}`);
    });
  }

  /**
   * Mendapatkan seluruh data dalam bentuk array
   * @returns {string[]}
   */
  getAllData() {
    return [...this._data]; // Return copy of array untuk menghindari modifikasi luar
  }

  // Static method dan instance untuk singleton
  static getInstance() {
    if (!PusatDataSingleton._instance) {
      PusatDataSingleton._instance = new PusatDataSingleton();
    }
    return PusatDataSingleton._instance;
  }
}

// Export singleton getter saja
module.exports = PusatDataSingleton;

```

Penjelasan:

Kode kelas PusatDataSingleton ini mengimplementasikan pola singleton untuk memastikan hanya ada satu instance yang menyimpan data dalam array privat _data. Kelas ini menyediakan metode untuk menambah data (addData), menghapus data berdasarkan indeks dengan validasi (removeData), mencetak semua data ke konsol (printAllData), dan mengambil salinan data lengkap (getAllData). Instance tunggal diakses melalui static method getInstance(), yang membuat instance baru hanya jika belum ada sebelumnya. Pendekatan ini menjaga konsistensi data di seluruh aplikasi dan mempermudah pengelolaan data secara terpusat dengan struktur kode yang rapi dan mudah dipahami.

Main.js

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

// Dapatkan instance singleton
const data1 = PusatDataSingleton.getInstance();
const data2 = PusatDataSingleton.getInstance();

// Tambahkan beberapa data
data1.addData("Muhammad Fathi Rafa");
data1.addData("Eduardo Bagus");
data1.addData("Mahrus Ali");
data1.addData("Asisten: Kak Andi");

// Cetak data dari instance data2 (seharusnya sama dengan data1)
console.log("Cetak dari data2:");
data2.printAllData();

// Hapus data pada indeks ke-3 (item ke-4)
data2.removeData(3);

console.log("\nCetak dari data1 setelah penghapusan:");
data1.printAllData();

// Cetak jumlah data yang ada
console.log(`\nJumlah data di data1: ${data1.getAllData().length}`);
console.log(`Jumlah data di data2: ${data2.getAllData().length}`);

```

Penjelasan:

Kode ini menunjukkan penggunaan singleton PusatDataSingleton dengan mengambil dua variabel data1 dan data2 yang sebenarnya merujuk pada instance yang sama. Data ditambahkan melalui data1, kemudian dicetak menggunakan data2 untuk menegaskan bahwa keduanya berbagi data yang sama. Setelah itu, data pada indeks ke-3 dihapus melalui data2, dan perubahan tersebut juga tercermin saat mencetak menggunakan data1. Akhirnya, jumlah data yang tersisa diperiksa dari kedua variabel, yang mengonfirmasi bahwa data terpusat dan konsisten di seluruh penggunaan instance singleton tersebut.

Output:

```
PS D:\KPL_MuhammadFathiRafa_2311104022_SE0701\14_Clean_Code\Jurnal_14_2311104022> node .\main.js
Cetak dari data2:
[0] Muhammad Fathi Rafa
[1] Eduardo Bagus
[2] Mahrus Ali
[3] Asisten: Kak Andi

Cetak dari data1 setelah penghapusan:
[0] Muhammad Fathi Rafa
[1] Eduardo Bagus
[2] Mahrus Ali

Jumlah data di data1: 3
Jumlah data di data2: 3
```

Perbedaan sebelum dan sesudah:

1. Naming Convention (Penamaan)
   Sebelumnya:
   Method dan variabel memakai campuran nama seperti GetDataSingleton, AddSebuahData, PrintSemuaData yang cenderung menggunakan PascalCase atau campuran bahasa Indonesia dan Inggris.
   Sekarang:
   Semua nama method dan variabel menggunakan camelCase yang merupakan standar di JavaScript, misalnya: getInstance(), addData(), printAllData().
   Penamaan ini membuat kode lebih konsisten dan mudah dimengerti oleh programmer JS pada umumnya.

2. Struktur Kode & White Space
   Sebelumnya:
   Struktur kode agak padat, indentation kurang rapi, dan spasi kadang tidak konsisten.
   Kode dibuat rapi, indentation konsisten 2 spasi, spasi antar blok dan parameter juga sudah rapih.

   3. Deklarasi Variabel dan Property
   Sebelumnya:
   Tidak jelas deklarasi atribut data (apakah lokal atau properti), kemungkinan menggunakan variabel global atau langsung di instance tanpa konvensi apapun.
   Sekarang:
   Data disimpan di properti instance dengan nama _data, memberikan tanda bahwa itu internal/private (konvensi awalan underscore).
   Variabel yang tidak berubah dideklarasikan const.

4. Penggunaan Singleton dengan Static Method
Sebelumnya:
Singleton diakses dengan GetDataSingleton() tanpa penjelasan bahwa itu static method.
Sekarang:
Singleton diakses dengan static method getInstance(), mengikuti pola yang sangat umum di JS dan bahasa lain untuk singleton.

5. Penanganan Error & Validasi
Sebelumnya:
Saat menghapus data (HapusSebuahData(3)), tidak ada pengecekan apakah indeks valid, berpotensi error atau silent fail.
Sekarang:
Ada validasi indeks sebelum menghapus, jika indeks salah maka muncul peringatan (console.warn).
6. Penggunaan Comments dan Dokumentasi
Sebelumnya:
Hampir tidak ada komentar yang menjelaskan fungsi kode.
Sekarang:
Ada komentar singkat dan JSDoc pada setiap method untuk menjelaskan apa fungsinya dan parameter yang digunakan.

7. Konsistensi Bahasa
Sebelumnya:
Nama method dan variabel menggunakan bahasa campuran Indonesia dan Inggris, misalnya AddSebuahData.
Sekarang:
Menggunakan bahasa Inggris (standard JS) agar lebih universal dan mudah dimengerti oleh developer global.