Tugas Jurnal Modul 14

1. Refactoring Standar Code naming convention, white space, variable, dan comments

```
// PusatDataSingleton.js

/**
 * Singleton class untuk menyimpan dan mengelola data bersama.
 */
class PusatDataSingleton {
  constructor() {
    // Cek apakah instance sudah ada, jika ya kembalikan instance yang sama
    if (PusatDataSingleton._instance) {
      return PusatDataSingleton._instance;
    }

    // Inisialisasi array penyimpanan data
    this.dataTersimpan = [];

    // Simpan instance agar hanya ada satu
    PusatDataSingleton._instance = this;
  }

  /**
   * Mengembalikan instance tunggal dari PusatDataSingleton.
   * @returns {PusatDataSingleton}
   */
  static getInstance() {
    if (!PusatDataSingleton._instance) {
      PusatDataSingleton._instance = new PusatDataSingleton();
    }
    return PusatDataSingleton._instance;
  }

  /**
   * Mengambil seluruh data yang tersimpan.
   * @returns {Array}
   */
  getSemuaData() {
    return this.dataTersimpan;
  }

  /**
   * Menampilkan seluruh data ke konsol.
   */
  printSemuaData() {
    console.log("Isi Data:");
    this.dataTersimpan.forEach((item, index) => {
      console.log(`${index + 1}. ${item}`);
    });
  }

  /**
   * Menambahkan sebuah data ke dalam penyimpanan.
   * @param {string} input - Data yang akan ditambahkan.
   */
  tambahData(input) {
    this.dataTersimpan.push(input);
  }

  /**
   * Menghapus sebuah data berdasarkan index.
   * @param {number} index - Index dari data yang akan dihapus.
   */
  hapusData(index) {
    if (index >= 0 && index < this.dataTersimpan.length) {
      this.dataTersimpan.splice(index, 1);
    } else {
      console.log("Index tidak valid.");
    }
  }
}

module.exports = PusatDataSingleton;
```

```
// main.js

const PusatDataSingleton = require("./PusatDataSingleton");

// Ambil instance dari singleton
const dataPertama = PusatDataSingleton.getInstance();
const dataKedua = PusatDataSingleton.getInstance();

// Tambahkan data melalui instance pertama
dataPertama.tambahData("Nama Anggota 1");
dataPertama.tambahData("Nama Anggota 2");
dataPertama.tambahData("Nama Asisten Praktikum");

// Cetak data dari instance kedua
console.log("\nCetak dari dataKedua:");
dataKedua.printSemuaData();

// Hapus data terakhir dari instance kedua
dataKedua.hapusData(2);

// Cetak hasil setelah penghapusan melalui instance pertama
console.log("\nSetelah penghapusan (dari dataPertama):");
dataPertama.printSemuaData();

// Tampilkan jumlah data di kedua instance
console.log("\nJumlah data:");
console.log("DataPertama count:", dataPertama.getSemuaData().length);
console.log("DataKedua count:", dataKedua.getSemuaData().length);
```

Refactoring pada kedua file dilakukan untuk meningkatkan keterbacaan, konsistensi, dan kesesuaian dengan standar penulisan kode yang baik. Pada file PusatDataSingleton.js, dilakukan perubahan pada penamaan class, method, dan atribut menggunakan naming convention camelCase dan PascalCase yang sesuai, serta penambahan komentar deskriptif di setiap method untuk memudahkan pemahaman. Struktur indentasi dirapikan menggunakan 2 spasi konsisten, dan atribut internal seperti dataTersimpan dideklarasikan dalam konstruktor dengan akses yang jelas. Sedangkan pada file main.js, penamaan variabel dibuat lebih deskriptif (dataPertama, dataKedua), struktur pemanggilan method diperjelas, dan penggunaan instance diperlihatkan secara eksplisit untuk menegaskan konsep Singleton. Refactoring ini menjadikan kode lebih profesional, mudah dipelihara, dan sesuai dengan pedoman Jurnal Modul 14.