LAPORAN PRAKTIKUM JURNAL MODUL 13

NIM / Nama: 2311104072 – Jauhar Fajar Zuhair

Bagian I: Pendahuluan

Laporan Jurnal Modul 13 ini membahas implementasi dan penggunaan **Singleton Design Pattern** dalam C#. Singleton adalah pola desain creational yang memastikan bahwa sebuah kelas hanya memiliki satu instance (objek) dan menyediakan titik akses global ke instance tersebut. Dalam praktikum ini, sebuah kelas `PusatDataSingleton` dibuat untuk mengelola sebuah daftar data string bersama (`DataTersimpan`), dan sebuah aplikasi konsol (`Program.cs`) digunakan untuk mendemonstrasikan cara kerja dan properti unik dari pola Singleton.

Bagian II: Implementasi Singleton (`PusatDataSingleton.cs`)

File ini berisi definisi kelas `PusatDataSingleton` yang menerapkan Singleton Pattern untuk mengelola data terpusat.

Struktur Kelas `PusatDataSingleton`:

```
public class PusatDataSingleton
    {
        // 1. Variabel statis privat untuk menyimpan satu-satunya instance
        private static PusatDataSingleton _instance = null;

        // 2. Objek untuk locking (thread safety saat inisialisasi pertama)
        private static readonly object _lock = new object();

        // 3. Properti publik untuk menyimpan data bersama (List of strings)
        //    Setter dibuat private agar list tidak bisa diganti dari luar,
        //    tapi isinya masih bisa dimodifikasi (via Add, Remove, dll.)
        public List<string> DataTersimpan { get; private set; }

        // 4. Konstruktor privat: Mencegah pembuatan instance dari luar kelas
        private PusatDataSingleton()
        {
            // Inisialisasi data saat instance pertama kali dibuat
            DataTersimpan = new List<string>();
            Console.WriteLine("Instance PusatDataSingleton dibuat."); //
Indikator pembuatan
        }

        // 5. Metode statis publik untuk mendapatkan instance tunggal
(Getter)
        public static PusatDataSingleton GetDataSingleton()
        {
```

```
            // Double-Checked Locking untuk thread safety dan efisiensi
            if (_instance == null) // Cek pertama (tanpa lock)
            {
                lock (_lock) // Kunci bagian kritis
                {
                    if (_instance == null) // Cek kedua (setelah lock)
                    {
                        // Buat instance HANYA jika belum ada
                        _instance = new PusatDataSingleton();
                    }
                }
            }
            // Kembalikan instance yang sudah ada atau yang baru dibuat
            return _instance;
        }

        // Metode untuk memanipulasi data tersimpan
        public void TambahData(string data)
        {
            DataTersimpan.Add(data);
            Console.WriteLine($"Data '{data}' ditambahkan.");
        }

        // Metode untuk melihat data tersimpan
        public void LihatData()
        {
            Console.WriteLine("\\\\n--- Data Tersimpan Saat Ini ---");
            if (DataTersimpan.Count == 0)
            {
                Console.WriteLine("(Kosong)");
            }
            else
            {
                foreach (string item in DataTersimpan)
                {
                    Console.WriteLine($"- {item}");
                }
            }
            Console.WriteLine("--------------------------------\\\\n");
        }
    }
}
```

Penjelasan Komponen Kunci Singleton:

1. **Private Static Instance (`_instance`):** Menyimpan satu-satunya objek `PusatDataSingleton`. Dimulai sebagai `null` (Lazy Initialization).
2. **Private Constructor:** `private PusatDataSingleton()` mencegah `new PusatDataSingleton()` dipanggil dari luar kelas. Ini adalah inti dari pembatasan instance. Di dalamnya, `DataTersimpan` diinisialisasi.
3. **Public Static Access Method (`GetDataSingleton()`):** Ini adalah satu-satunya cara untuk mendapatkan referensi ke objek Singleton. Metode ini:

    - Menggunakan *Double-Checked Locking* (`if` di luar dan di dalam `lock`) untuk memastikan *thread safety* saat instance pertama kali dibuat dalam lingkungan multi-threaded, sambil tetap efisien untuk akses berikutnya (tidak perlu `lock` jika `_instance` sudah ada).
    - Membuat instance (`_instance = new PusatDataSingleton();`) *hanya jika* `_instance` masih `null` di dalam blok `lock`.
    - Selalu mengembalikan referensi ke `_instance` yang sama.
4. **Shared State (`DataTersimpan`):** Properti `List<string>` yang menyimpan data. Karena hanya ada satu instance `PusatDataSingleton`, list ini juga hanya ada satu dan dibagikan oleh semua bagian program yang mengakses Singleton.
5. **Data Methods (`TambahData`, `LihatData`):** Metode biasa untuk berinteraksi dengan data bersama yang dikelola oleh Singleton.

Bagian III: Penggunaan Singleton (`Program.cs`)

Aplikasi konsol ini mendemonstrasikan bagaimana Singleton diakses dan bagaimana state-nya bersifat global dan tunggal.

Logika `Program.Main`:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== IMPLEMENTASI DESIGN PATTERN SINGLETON ===");
        Console.WriteLine();

        // A. Membuat dua variabel dengan tipe PusatDataSingleton
        //    Keduanya akan mendapatkan referensi ke instance yang SAMA
        Console.WriteLine("Mendapatkan instance data1...");
        PusatDataSingleton data1 = PusatDataSingleton.GetDataSingleton();

        Console.WriteLine("\\\\nMendapatkan instance data2...");
        PusatDataSingleton data2 = PusatDataSingleton.GetDataSingleton();

        // Verifikasi bahwa kedua variabel merujuk ke instance yang sama
        Console.WriteLine("\\\\nMemeriksa apakah data1 dan data2 adalah
instance yang sama...");
        if (ReferenceEquals(data1, data2))
        {
            Console.WriteLine(">>> Benar, data1 dan data2 merujuk ke instance
Singleton yang sama.");
        }
        else
        {
            Console.WriteLine(">>> Salah, data1 dan data2 merujuk ke instance
yang berbeda (Ini seharusnya tidak terjadi!).");
        }
        Console.WriteLine();
```

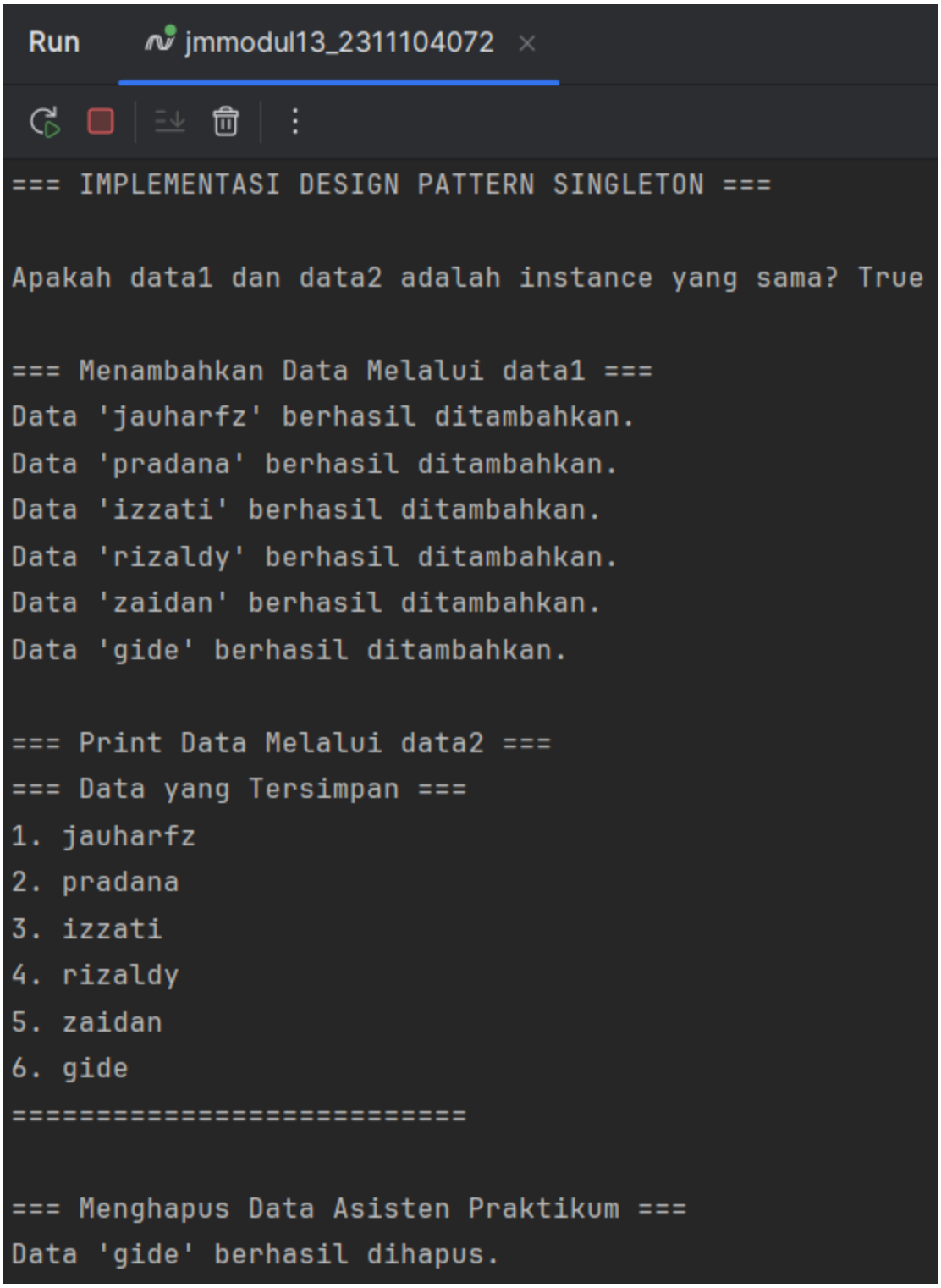
```
        // B. Menambahkan data melalui variabel data1
        data1.TambahData("Data Penting A");
        data1.TambahData("Konfigurasi X");

        // C. Menampilkan data melalui variabel data2
        //    Akan menampilkan data yang ditambahkan melalui data1,
        //    membuktikan state bersama.
        Console.WriteLine("\\\\nMelihat data melalui instance data2:");
        data2.LihatData();

        // D. Menambahkan data lagi melalui variabel data2
        data2.TambahData("Log Aktivitas Y");

        // E. Menampilkan data akhir melalui variabel data1
        Console.WriteLine("\\\\nMelihat data akhir melalui instance data1:");
        data1.LihatData();

        Console.WriteLine("=== DEMO SELESAI ===");
    }
}
}
```

Penjelasan Demonstrasi:

1. **Mendapatkan Instance:** `PusatDataSingleton.GetDataSingleton()` dipanggil dua kali untuk `data1` dan `data2`. Karena ini Singleton, kedua panggilan akan mengembalikan referensi ke objek yang *sama*. Pesan "Instance PusatDataSingleton dibuat." hanya akan muncul sekali (saat panggilan pertama).
2. **Verifikasi Instance:** `ReferenceEquals(data1, data2)` digunakan untuk membandingkan apakah kedua variabel menunjuk ke lokasi memori yang sama. Hasilnya harus `true`, membuktikan bahwa hanya ada satu instance.
3. **Manipulasi Data:** Data ditambahkan menggunakan `data1.TambahData()`.
4. **Akses Data Bersama:** Data kemudian dilihat menggunakan `data2.LihatData()`. Output akan mencerminkan data yang baru ditambahkan melalui `data1`, menunjukkan bahwa `DataTersimpan` adalah state yang sama untuk kedua variabel (karena mereka adalah objek yang sama).
5. **Manipulasi Lanjutan:** Data ditambahkan lagi melalui `data2`.
6. **Akses Data Akhir:** Data dilihat lagi melalui `data1`, menunjukkan semua data yang telah ditambahkan melalui kedua variabel.

Bagian IV: Hasil Eksekusi (Contoh Tampilan Konsol)

Output yang diharapkan dari eksekusi `Program.cs`:


Run
jmmodul13_2311104072
=== IMPLEMENTASI DESIGN PATTERN SINGLETON ===
Apakah data1 dan data2 adalah instance yang sama? True
=== Menambahkan Data Melalui data1 ===
Data 'jauharfz' berhasil ditambahkan.
Data 'pradana' berhasil ditambahkan.
Data 'izzati' berhasil ditambahkan.
Data 'rizaldy' berhasil ditambahkan.
Data 'zaidan' berhasil ditambahkan.
Data 'gide' berhasil ditambahkan.
=== Print Data Melalui data2 ===
=== Data yang Tersimpan ===
1. jauharfz
2. pradana
3. izzati
4. rizaldy
5. zaidan
6. gide
==========
=== Menghapus Data Asisten Praktikum ===
Data 'gide' berhasil dihapus.

```
=== Print Data Melalui data1 (Setelah Penghapusan) ===
=== Data yang Tersimpan ===
1. jauharfz
2. pradana
3. izzati
4. rizaldy
5. zaidan
============================

=== Jumlah Elemen dalam List ===
Jumlah elemen di data1: 5
Jumlah elemen di data2: 5

=== Demonstrasi Singleton Behavior ===
Data 'Data Baru dari data1' berhasil ditambahkan.
Data ditambahkan melalui data1, tapi akan terlihat di data2:
=== Data yang Tersimpan ===
1. jauharfz
2. pradana
3. izzati
4. rizaldy
5. zaidan
6. Data Baru dari data1
============================

Press any key to exit...
```

Penjelasan Hasil:

- Pesan "Instance PusatDataSingleton dibuat." hanya muncul sekali, saat `data1` pertama kali meminta instance. Panggilan untuk `data2` tidak membuat instance baru.
- Verifikasi `ReferenceEquals` mengonfirmasi bahwa `data1` dan `data2` adalah objek yang identik.

- Data yang ditambahkan melalui `data1` terlihat saat diakses melalui `data2`, dan sebaliknya. Ini membuktikan bahwa keduanya berbagi state (`DataTersimpan`) yang sama, yang merupakan karakteristik utama Singleton.
