
LAPORAN PRAKTIKUM JURNAL MODUL 7

NIM / Nama: 2311104072 – Jauhar Fajar Zuhair

Bagian I: Pendahuluan

Jurnal Modul 7 melanjutkan eksplorasi materi sebelumnya dengan fokus pada penerapan Git/GitHub untuk kolaborasi tim dan deserialisasi data dari format JSON ke dalam objek C#. Praktikum ini mencakup tiga tugas utama deserialisasi: membaca data detail mahasiswa, membaca daftar anggota tim, dan membaca entri glossary yang lebih kompleks. Selain itu, jurnal ini juga melibatkan alur kerja Git/GitHub standar seperti pembuatan branch untuk setiap fitur (tugas deserialisasi), commit perubahan, push ke remote repository, dan akhirnya membuat Pull Request untuk menggabungkan semua pekerjaan ke branch utama.

Bagian II: Konsep Utama

1. **JSON Deserialization (`System.Text.Json`):** Proses konversi data teks berformat JSON menjadi instance objek C#. Ini krusial untuk mengolah data yang diterima dari API web atau file konfigurasi. Jurnal ini menggunakan `JsonSerializer.Deserialize<T>()` untuk memetakan struktur JSON (termasuk objek bersarang dan array) ke kelas-kelas C# yang sesuai. Opsi seperti `PropertyNameCaseInsensitive = true` digunakan untuk menangani perbedaan penamaan antara properti JSON dan C#.
2. **Struktur Kelas C# untuk JSON:** Agar deserialisasi berhasil, struktur kelas C# (termasuk properti dan kelas bersarang/nested class) harus mencerminkan struktur data dalam file JSON. Jurnal ini memerlukan pembuatan beberapa set kelas untuk tiga file JSON yang berbeda, termasuk struktur yang cukup kompleks untuk data glossary.

Bagian III: Implementasi Kode

Implementasi jurnal ini melibatkan tiga kelas utama untuk deserialisasi dan satu kelas program utama:

1. **`DataMahasiswa2311104072.cs`**:
    - Bertujuan membaca file `jurnal7_1_2311104072.json` yang berisi detail data seorang mahasiswa, termasuk alamat dan daftar mata kuliah (menggunakan nested class `Address` dan `Course`).
    - Method `ReadJSON()` melakukan deserialisasi dan mencetak semua informasi mahasiswa.

```
public class DataMahasiswa_2311104072
{
    // Properti: firstName, lastName, gender, age, address (tipe
Address), courses (List<Course>)
    public string firstName { get; set; }
    // ... properti lainnya ...
    public Address address { get; set; }
    public List<Course> courses { get; set; }

    // Nested class Address { streetAddress, city, state }
    public class Address { /* ... properti ... */ }
    // Nested class Course { code, name }
    public class Course { /* ... properti ... */ }

    public void ReadJSON()
    {
        string jsonFilePath = "jurnal7_1_2311104072.json";
        Console.WriteLine($"\n--- Membaca Data Mahasiswa dari
{jsonFilePath} ---");
        try
        {
            string jsonData = File.ReadAllText(jsonFilePath);
            var options = new JsonSerializerOptions {
PropertyNameCaseInsensitive = true };
            DataMahasiswa_2311104072 data =
JsonSerializer.Deserialize<DataMahasiswa_2311104072>(jsonData,
options);

            if (data != null)
            {
                // Mencetak semua data (nama, gender, umur, alamat,
courses)
                Console.WriteLine($"Nama Lengkap: {data.firstName}
{data.lastName}");
                // ... print properti lainnya ...
                if (data.courses != null && data.courses.Count > 0) {
                    Console.WriteLine("Mata Kuliah:");
                    foreach (var course in data.courses) { /* print
course */ }
                }
            }
            // ... penanganan error ...
        }
        // ... catch Exception ...
    }
}
```

Penjelasan Snippet: Method `ReadJSON` membaca file JSON, melakukan deserialisasi dengan opsi case-insensitive, lalu mencetak semua properti objek `DataMahasiswa_2311104072`, termasuk data dari nested class `Address` dan iterasi melalui list `Course`.

2. **`TeamMembers2311104072.cs`**:
    - Bertujuan membaca file `jurnal7_2_2311104072.json` yang berisi daftar anggota tim dalam sebuah array JSON. Menggunakan nested class `Member`.
    - Method `ReadJSON()` mendeserialisasi data tim dan mencetaknya sesuai format yang diminta.

```
public class TeamMembers_2311104072
{
    // Properti utama: List<Member>
    public List<Member> members { get; set; }

    // Nested class Member { firstName, lastName, gender, age, nim }
    public class Member { /* ... properti ... */ }

    public void ReadJSON()
    {
        string jsonFilePath = "jurnal7_2_2311104072.json";
        Console.WriteLine($"\n--- Membaca Data Anggota Tim dari
{jsonFilePath} ---");
        try
        {
            string jsonData = File.ReadAllText(jsonFilePath);
            var options = new JsonSerializerOptions {
PropertyNameCaseInsensitive = true };
            // Deserialisasi objek root yang memiliki properti
'members'
            TeamMembers_2311104072 teamData =
JsonSerializer.Deserialize<TeamMembers_2311104072>(jsonData, options);

            if (teamData?.members != null && teamData.members.Count >
0)
            {
                Console.WriteLine("Team member list:");
                foreach (var member in teamData.members)
                {
                    // Mencetak sesuai format: "<nim> <firstname>
<lastname> (<age> <gender>)"
                    Console.WriteLine($"{member.nim} {member.firstName}
{member.lastName} ({member.age} {member.gender})");
                }
            }
            // ... penanganan error ...
        }
        // ... catch Exception ...
    }
}
```

Penjelasan Snippet: Method ini membaca file JSON yang strukturnya adalah objek dengan satu properti `members` (berupa array). Setelah deserialisasi, ia mengiterasi list `members` dan mencetak detail setiap anggota sesuai format yang ditentukan.

3. **`GlossaryItem2311104072.cs`**:
    - Bertujuan membaca file `jurnal7_3_2311104072.json` yang memiliki struktur JSON sangat bersarang (nested). Kelas-kelas C# (`GlossaryRoot_2311104072`, `Glossary`, `GlossDiv`, `GlossList`, `GlossEntry`, `GlossDef`) dibuat untuk mencerminkan struktur ini.
    - Method `ReadJSON()` *ditempatkan di kelas terpisah* `GlossaryItem_2311104072` (sesuai instruksi 6B). Method ini mendeserialisasi seluruh struktur, namun hanya mencetak detail dari bagian `GlossEntry`.

```
public class GlossEntry { /* ID, SortAs, GlossTerm, Acronym, Abbrev,
GlossDef, GlossSee */ }
public class GlossDef { /* para, GlossSeeAlso (List<string>) */ }
// ... dan seterusnya untuk kelas parent ...
public class GlossaryRoot_2311104072 { public Glossary glossary { get;
set; } }


// Kelas terpisah sesuai instruksi 6B
public class GlossaryItem_2311104072
{
    public void ReadJSON()
    {
        string jsonFilePath = "jurnal7_3_2311104072.json";
        Console.WriteLine($"\n--- Membaca Data Glossary dari
{jsonFilePath} ---");
        try
        {
            string jsonData = File.ReadAllText(jsonFilePath);
            var options = new JsonSerializerOptions {
PropertyNameCaseInsensitive = true };
            GlossaryRoot_2311104072 root =
JsonSerializer.Deserialize<GlossaryRoot_2311104072>(jsonData, options);

            // Akses bagian GlossEntry (Instruksi 6D: print GlossEntry
saja)
            // Menggunakan null-conditional operator (?.) untuk
keamanan
            GlossEntry entry =
root?.glossary?.GlossDiv?.GlossList?.GlossEntry;

            if (entry != null)
            {
                Console.WriteLine("GlossEntry Details:");
                Console.WriteLine($"  ID: {entry.ID}");
                Console.WriteLine($"  SortAs: {entry.SortAs}");
                Console.WriteLine($"  GlossTerm: {entry.GlossTerm}");
                Console.WriteLine($"  Acronym: {entry.Acronym}");
                Console.WriteLine($"  Abbrev: {entry.Abbrev}");
                if (entry.GlossDef != null)
                {
```

```
                    Console.WriteLine("  GlossDef:");
                    Console.WriteLine($"    para:
{entry.GlossDef.para}");
                    if (entry.GlossDef.GlossSeeAlso != null &&
entry.GlossDef.GlossSeeAlso.Count > 0) {
                        Console.WriteLine($"    GlossSeeAlso:
{string.Join(", ", entry.GlossDef.GlossSeeAlso)}");
                    }
                }
                Console.WriteLine($"  GlossSee: {entry.GlossSee}");
            }
            // ... penanganan error jika entry null ...
        }
        // ... catch Exception ...
    }
}
```

Penjelasan Snippet: Method ini menunjukkan deserialisasi struktur JSON yang kompleks. Setelah mendapatkan objek `root`, ia menavigasi ke bawah melalui properti-properti bersarang (`glossary`, `GlossDiv`, `GlossList`) untuk mendapatkan objek `GlossEntry`. Kemudian, hanya properti dari `GlossEntry` (dan `GlossDef` di dalamnya) yang dicetak.

4. **`Program.cs`**:
    - Titik masuk aplikasi (`Main` method).
    - Membuat instance dari ketiga kelas (`DataMahasiswa...`, `TeamMembers...`, `GlossaryItem...`) dan memanggil method `ReadJSON()` masing-masing secara berurutan untuk menampilkan semua hasil deserialisasi.

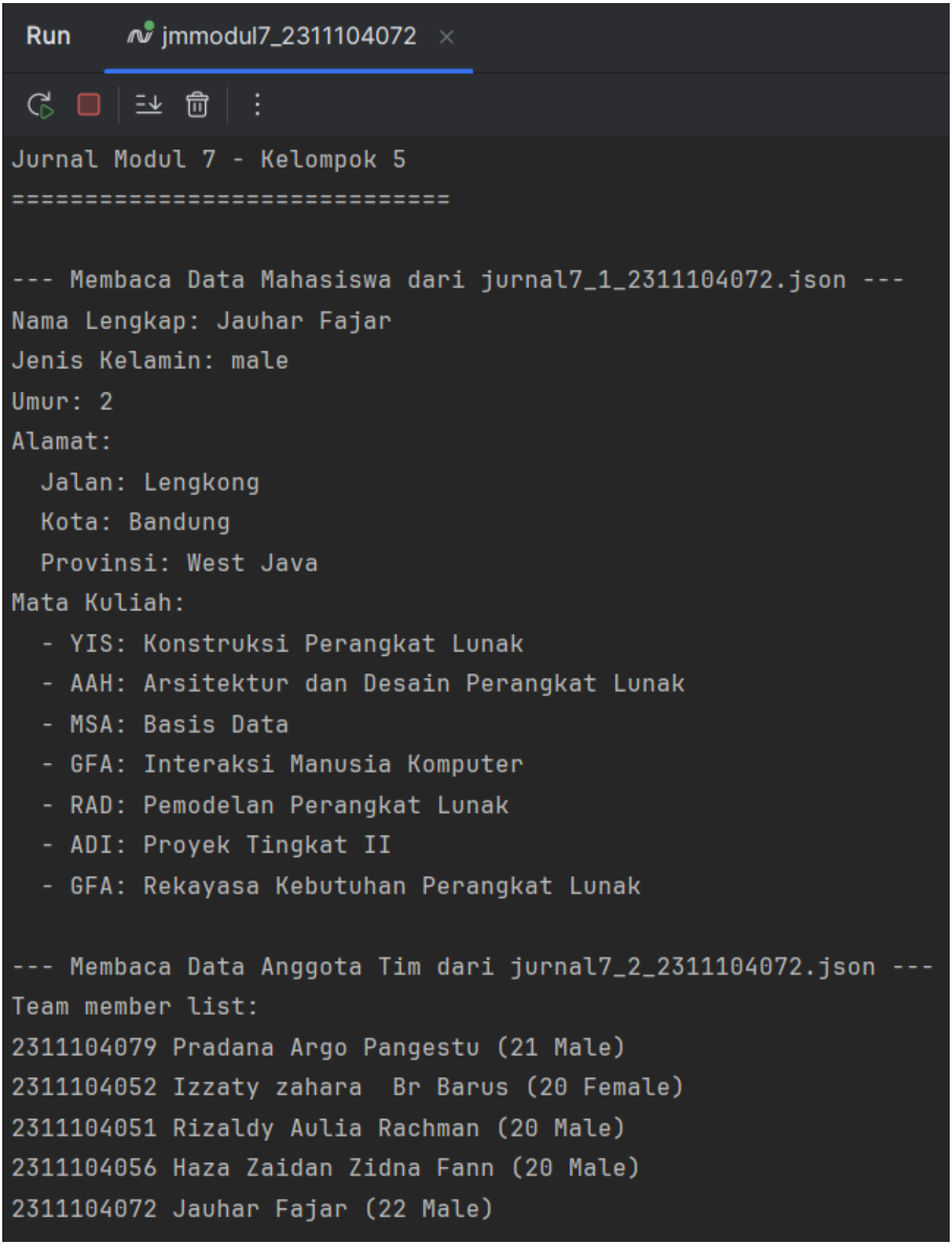
```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Jurnal Modul 7 - Kelompok XY"); // Ganti XY
dengan nomor kelompok
        Console.WriteLine("================================");

        // Panggil ReadJSON untuk DataMahasiswa
        DataMahasiswa_2311104072 dataMhs = new
DataMahasiswa_2311104072();
        dataMhs.ReadJSON();

        // Panggil ReadJSON untuk TeamMembers
        TeamMembers_2311104072 team = new TeamMembers_2311104072();
        team.ReadJSON();

        // Panggil ReadJSON untuk GlossaryItem
        GlossaryItem_2311104072 glossary = new
GlossaryItem_2311104072();
        glossary.ReadJSON();

        Console.WriteLine("\n================================");
        Console.WriteLine("Selesai.");
```

```
        }
}
```

Penjelasan Snippet: Kode `Main` sangat sederhana, hanya menginisialisasi objek dari setiap kelas tugas dan memanggil method `ReadJSON` mereka untuk menjalankan logika pembacaan dan pencetakan data.

Bagian IV: Hasil Eksekusi (Contoh Tampilan Konsol)

Output program di konsol akan menampilkan hasil dari ketiga proses deserialisasi secara berurutan:


Run
jmmodul7_2311104072
Jurnal Modul 7 - Kelompok 5
==========
--- Membaca Data Mahasiswa dari jurnal7_1_2311104072.json ---
Nama Lengkap: Jauhar Fajar
Jenis Kelamin: male
Umur: 2
Alamat:
Jalan: Lengkong
Kota: Bandung
Provinsi: West Java
Mata Kuliah:
- YIS: Konstruksi Perangkat Lunak
- AAH: Arsitektur dan Desain Perangkat Lunak
- MSA: Basis Data
- GFA: Interaksi Manusia Komputer
- RAD: Pemodelan Perangkat Lunak
- ADI: Proyek Tingkat II
- GFA: Rekayasa Kebutuhan Perangkat Lunak
--- Membaca Data Anggota Tim dari jurnal7_2_2311104072.json ---
Team member list:
2311104079 Pradana Argo Pangestu (21 Male)
2311104052 Izzaty zahara Br Barus (20 Female)
2311104051 Rizaldy Aulia Rachman (20 Male)
2311104056 Haza Zaidan Zidna Fann (20 Male)
2311104072 Jauhar Fajar (22 Male)

```
--- Membaca Data Glossary dari jurnal7_3_2311104072.json ---
GlossEntry Details:
  ID: SGML
  SortAs: SGML
  GlossTerm: Standard Generalized Markup Language
  Acronym: SGML
  Abbrev: ISO 8879:1986
  GlossDef:
    para: A meta-markup language, used to create markup languages such as DocBook.
    GlossSeeAlso: GML, XML
  GlossSee: markup

Tekan tombol apapun untuk keluar...
```

Penjelasan Contoh Output: Output menunjukkan data mahasiswa, diikuti daftar anggota tim dengan format spesifik, dan terakhir detail entri glossary (hanya bagian `GlossEntry`). Setiap bagian diawali header yang menandakan file mana yang sedang dibaca.