

*Berdo'alah sebelum mengerjakan. Dilarang berbuat curang.
Tugas ini untuk mengukur kemampuan anda, jadi kerjakan dengan sepenuh hati.
Selamat belajar, semoga sukses !*

| | | |
|--|---|------------------------|
| Nama Mahasiswa: Fadillah Rizky R | NIM: 1301164493 | Nilai: |
| Nama Mahasiswa: Mazaya Z D | NIM: 1301154508 | Nilai: |
| Nama Mahasiswa: Renaning Karutami Susilo | NIM: 1301154466 | Nilai: |

Siapkan tools berikut sebelum mengerjakan:

1. Go Programming Language (<https://golang.org/dl/>).
2. Visual Studio Code (<https://code.visualstudio.com/>) atau LiteIDE (<https://github.com/visualfc/liteide>).
3. Disarankan untuk menggunakan linux dengan distro fedora (<https://getfedora.org/id/workstation/>).
4. Buatlah git repository pada <https://github.com/> kemudian push semua kode dan hasil laporan anda ke dalam repository github yang sudah anda buat. Kumpulkan link repository github tersebut sebagai tanda bahwa anda mengerjakan tugas modul ini.
5. Lakukan instalasi flatbuffer (<https://google.github.io/flatbuffers/>) untuk mengerjakan salah satu tugas pada modul ini.

| | | |
|-------|------|--------|
| Nama: | NIM: | Nilai: |
|-------|------|--------|

Soal No 1 (JSON Marshal)

```
package main

import (
    "encoding/json"
    "fmt"
)

type Person struct {
    FirstName string `json:"firstName"`
    LastName  string `json:"lastName"`
}

func main() {
    bytes, err := json.Marshal(Person{
        FirstName: "John",
        LastName:  "Dow",
    })
    if err != nil {
        panic(err)
    }

    fmt.Println(string(bytes))
}
```

Jalankan program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

```
[mazayaazd@localhost src]$ go run jsonmarshal.go
{"firstName":"John","lastName":"Dow"}
[mazayaazd@localhost src]$
```

Fungsi *json.Marshal* digunakan untuk decoding data ke json. Data tersebut bisa berupa variable objek cetakan struct `map[string]interface{}`, bisa juga bertipe array.

Program pada nomor 1 adalah contoh cara encode data ke bentuk json.

- Pertama import package yang dibutuhkan dan siapkan struct *Person*.
- Hasil encode nantinya akan disimpan ke variable objek cetakan struct *Person*.
- Buat contoh struct *Person*.
- Buat json dari contoh data
- Buat pesan ketika error
- Hasil encode adalah bertipe `[]byte`. Casting ke *string* bisa digunakan untuk menampilkan data.

| | | |
|-------|------|--------|
| Nama: | NIM: | Nilai: |
|-------|------|--------|

Soal No 2 (JSON Unmarshal)

```
package main

import (
    "encoding/json"
    "fmt"
)

type Person struct {
    FirstName string `json:"firstName"`
    LastName  string `json:"lastName"`
}

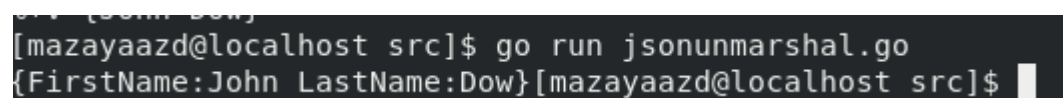
func main() {
    in := `{"firstName":"John","lastName":"Dow"}`
    bytes := []byte(in)

    var p Person
    err := json.Unmarshal(bytes, &p)
    if err != nil {
        panic(err)
    }

    fmt.Printf("%+v", p)
}
```

Jalankan program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:



```
[mazayaazd@localhost src]$ go run jsonunmarshal.go
{FirstName:John LastName:Dow}[mazayaazd@localhost src]$
```

Data json tipenya adalah `[]byte`, bisa didapat dari file ataupun string (dengan hasil casting). Dengan menggunakan `json.Unmarshal`, data tersebut bisa dikonversi menjadi bentuk objek, seperti bentuk `map[string]interface{}` ataupun variable objek hasil struct.

Program pada nomor 2 adalah contoh cara decoding json ke bentuk objek.

- Pertama import package yang dibutuhkan dan siapkan struct *Person*.
- Hasil decode nantinya akan disimpan ke variable objek cetakan struct *Person*.
- Selanjutnya siapkan data json string sederhana, gunakan casting ke `[]byte` agar dideteksi sebagai data json.
- Dalam penggunaan fungsi `json.Unmarshal`, variable yang akan menampung hasil decode harus di-passing sebagai pointer (`&p`).
- Pada kode di soal nomor 2 bisa dilihat bahwa terdapat property struct *Person* yang memiliki tag, yaitu *FirstName* dan *LastName*. Tag tersebut digunakan untuk mapping data json ke property yang bersangkutan.
- Data json yang akan diparsing memiliki 2 property yaitu *FirstName* dan *LastName*.
- Property *FirstName* struct tersebut kemudian ditugaskan untuk menampung data json property *firstname*, ditandai dengan tag `'json:"firstname"'` pada saat deklarasi structnya.

| | | |
|-------|------|--------|
| Nama: | NIM: | Nilai: |
|-------|------|--------|

Soal No 3 (Flatbuffer dan Protocol Buffer)

Jalankan program pada repository github berikut: <https://github.com/jonog/grpc-flatbuffers-example>

Berikan analisis berupa:

1. Apakah outputnya (berikan printscreen)!
2. Jelaskan cara kerjanya dan buatlah diagram FSMnya!
3. Analisis perbedaan dari protocol buffer dan flatbuffer!

Jawaban:

1. Output
2. Cara kerja
3. Protokol Buffer memang relatif mirip dengan FlatBuffers, dengan perbedaan utama adalah bahwa FlatBuffers tidak memerlukan langkah parsing / membongkar untuk representasi sekunder sebelum dapat mengakses data, sering ditambah dengan alokasi memori per-objek. Kode urutan ukurannya jauh lebih besar. Protokol Buffer tidak memiliki impor / ekspor teks opsional atau fitur bahasa skema seperti penyatuan.