

Nama : Yuwantoro M.
NIM : 1301150042
Kelas : TELE – 40 - GAB

1. Diagram TCP Finite State Machine.

Diagram ini adalah diagram yang menjelaskan alur terjadinya three way handshake pada saat pembuatan koneksi TCP. Starting point berada pada state CLOSED ketika belum ada koneksi.

➤ Client

1. Client mengirim SYN (SYN SENT)
2. Client menerima SYN dan ACK, kemudian mengirim ACK (CONNECTION ESTABLISHED)
3. Client mengirimkan FIN (FIN WAIT 1)
4. Client menerima ACK (FIN WAIT 2)
5. Client menerima FIN dan mengirim ACK (TIME WAIT)
6. Terjadi timeout, maka koneksi berakhir (CLOSED)

➤ Server

1. Server membuat koneksi secara passive open (LISTEN)
2. Server menerima SYN, kemudian mengirim SYN, ACK (SYN RCVD)
3. Server menerima ACK (CONNECTION ESTABLISHED)
4. Server menerima FIN, kemudian mengirimkan ACK (CLOSE WAIT)
5. Server mengirim FIN (LAST ACK)
6. Server mengakhiri koneksi (CLOSED)

2.

The screenshot shows the Visual Studio Code editor with a file named `for.go`. The code defines a `main` function that uses a `for` loop to print numbers 1 through 3. The terminal output shows the command `go run for.go` being executed, resulting in the numbers 1, 2, and 3 being printed on separate lines.

```

1 package main
2 import "fmt"
3
4 func main() {
5     i := 1
6     for i <= 3 {
7         fmt.Println(i)
8         i = i + 1
9     }
10 }

```

```

[yuwantoro@localhost ~]$ cd
[yuwantoro@localhost ~]$ cd Documents
[yuwantoro@localhost Documents]$ go run for.go
1
2
3

```

- Perulangan for merupakan instruksi untuk melakukan perulangan sesuai dengan kondisi yang ditentukan
- Perulangan for dapat berhenti apabila kondisi sudah false, atau jika ada break
- Perulangan for juga dapat di-skip dengan menggunakan continue

The screenshot shows the Visual Studio Code editor with a file named `if-else.go`. The code defines a `main` function that uses `if` and `else` statements to check if 7 is even or odd, and if 8 is divisible by 4. The terminal output shows the command `go run if-else.go` being executed, resulting in the following output: `7 is odd`, `8 is divisible by 4`, and `9 has 1 digit`.

```

1 package main
2
3 import "fmt"
4
5 func main() {
6     if 7%2 == 0 {
7         fmt.Println("7 is even")
8     } else {
9         fmt.Println("7 is odd")
10    }
11
12    if 8%4 == 0 {
13        fmt.Println("8 is divisible by 4")
14    }
15 }

```

```

[yuwantoro@localhost ~]$ cd Documents
[yuwantoro@localhost Documents]$ go run if-else.go
7 is odd
8 is divisible by 4
9 has 1 digit

```

- If akan dijalankan jika kondisinya true
- Jika kondisi if false, maka akan berlanjut ke else if setelahnya
- Jika semua if kondisinya false, maka yang dijalankan adalah else

3.

The screenshot shows the Visual Studio Code editor with the file `array.go` open. The code defines a `main` function that creates an array `a` of 5 integers, prints its initial state, sets the value at index 4 to 100, prints the updated state, and prints the length of the array. The terminal output shows the execution of `go run array.go`, which produces the following output:

```
[yuwantoro@localhost Documents]$ go run array.go
emp: [0 0 0 0 0]
set: [0 0 0 0 100]
get: 100
len: 5
dcl [1 2 3 4 5]
2d: [[0 1 2] [1 2 3]]
[yuwantoro@localhost Documents]$
```

- Array merupakan sekumpulan variabel yang memiliki tipe data yang sama
- Array mempunyai indeks untuk menentukan nilainya
- Panjang dari suatu array dapat diketahui dengan fungsi `len(variabelArray)`
- Array bisa berbentuk 2 dimensi atau lebih

The screenshot shows the Visual Studio Code editor with the file `function.go` open. The code defines two functions: `plus` which takes two integers and returns their sum, and `plusPlus` which takes three integers and returns their sum. The `main` function calls `plus` with arguments 1 and 2. The terminal output shows the execution of `go run function.go`, which produces the following output:

```
[yuwantoro@localhost ~]$ cd Documents
[yuwantoro@localhost Documents]$ go run function.go
1+2 = 3
1+2+3 = 6
[yuwantoro@localhost Documents]$
```

- Function adalah sekumpulan pernyataan yang akan dijalankan jika namanya dipanggil
- Function dapat mengembalikan bermacam-macam tipe data

4.

```

struct.go - Visual Studio Code
File Edit Selection View Go Debug Terminal Help

home > yuwantoro > Documents > struct.go
1 package main
2
3 import "fmt"
4
5 type person struct {
6     name string
7     age  int
8 }
9
10 func main() {
11     fmt.Println(person{"Bob", 20})
12
13     fmt.Println(person{name: "Alice", age: 30})
14 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
{Bob 20}
{Alice 30}
{Fred 0}
&{Ann 40}
Sean
50
51
[yuwantoro@localhost Documents]$

```

- Struct adalah instruksi untuk membuat tipe data bentukan
- Sebuah struct bisa mempunyai berbagai variabel yang tipe datanya berbeda
- Jika pada saat pemanggilan struct terdapat variabel yang tidak didefinisikan nilainya, maka nilainya menjadi 0 atau nil

```

method.go - Visual Studio Code
File Edit Selection View Go Debug Terminal Help

home > yuwantoro > Documents > method.go
1 package main
2
3 import "fmt"
4
5 type rect struct {
6     width, height int
7 }
8
9 func (r *rect) area() int {
10     return r.width * r.height
11 }
12
13 func (r rect) perim() int {
14     return 2*r.width + 2*r.height
15 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
[yuwantoro@localhost ~]$ cd Documents
[yuwantoro@localhost Documents]$ go run method.go
area: 50
perim: 30
area: 50
perim: 30
[yuwantoro@localhost Documents]$

```

- Method adalah fungsi yang memiliki akses ke properti struct
- Perbedaannya dengan fungsi, pada saat deklarasi method ditentukan struct dari method tersebut
- Untuk memanggilnya juga harus diawali dengan variabel struct terlebih dahulu

5.

```

multipereturnvalue.go - Visual Studio Code
File Edit Selection View Go Debug Terminal Help

array.go function.go struct.go method.go multipereturnvalue.go x

home > yuwantoro > Documents > multipereturnvalue.go
1 package main
2
3 import "fmt"
4
5 func vals() (int, int) {
6     return 3, 7
7 }
8
9 func main() {
10     a, b := vals()
11     fmt.Println(a)
12     fmt.Println(b)
13
14     , c := vals()
15 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
[yuwantoro@localhost ~]$ cd Documents
[yuwantoro@localhost Documents]$ go run multipereturnvalue.go
3
7
[yuwantoro@localhost Documents]$

```

- Pada bahasa Go, fungsi dapat mengembalikan lebih dari 1 nilai

```

commandline.go - Visual Studio Code
File Edit Selection View Go Debug Terminal Help

function.go struct.go method.go multipereturnvalue.go commandline.go x

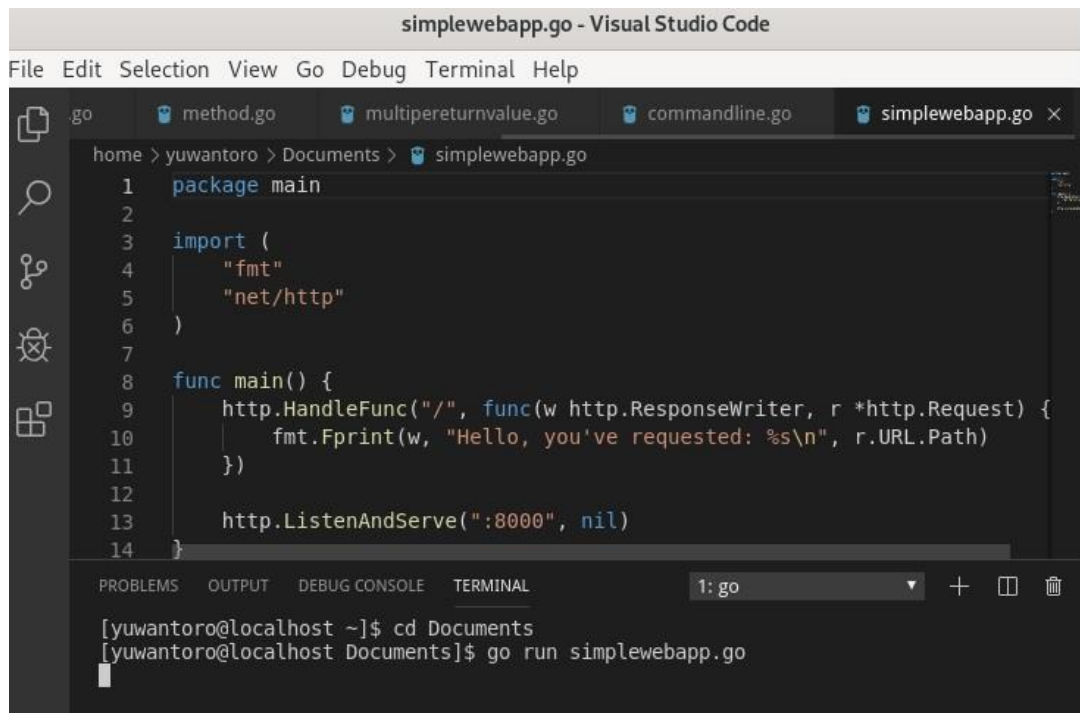
home > yuwantoro > Documents > commandline.go
1 package main
2
3 import (
4     "flag"
5     "fmt"
6 )
7
8 func main() {
9     wordPtr := flag.String("word", "foo", "a string")
10
11     numbPtr := flag.Int("numb", 42, "an int")
12     boolPtr := flag.Bool("fork", false, "a bool")
13
14     var svar string
15 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
[yuwantoro@localhost ~]$ cd Documents
[yuwantoro@localhost Documents]$ go run commandline.go
word: foo
numb: 42
fork false
svar: bar
tail: []
[yuwantoro@localhost Documents]$

```

- Deklarasi flag berupa string, integer, dan boolean.
- Flag dideklarasikan dengan assign ke variable

6.



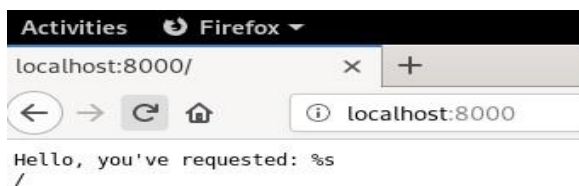
The screenshot shows the Visual Studio Code editor with a Go file named `simplewebapp.go` open. The file is located at `home > yuwantoro > Documents > simplewebapp.go`. The code defines a `main` package and a `main` function. The `main` function imports `fmt` and `net/http` packages. It uses `http.HandleFunc` to register a handler for the root path `/`, which prints a message to the log. Finally, it calls `http.ListenAndServe` to start the web server on port 8000.

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func main() {
9     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
10         fmt.Fprint(w, "Hello, you've requested: %s\n", r.URL.Path)
11     })
12
13     http.ListenAndServe(":8000", nil)
14 }
```

Below the editor, the terminal shows the command `go run simplewebapp.go` being executed in the `Documents` directory.

- Go bisa digunakan untuk membuat aplikasi web
- `HandleFunc` berfungsi untuk menentukan route dan menentukan konten web ketika alamat diakses
- `ListenAndServe` berfungsi menentukan port dan menjalankan aplikasi pada port yang ditentukan

Hasil program ketika dibuka dengan localhost di browser.



7.

The screenshot shows the Visual Studio Code interface with a Go file named `configfile.go` open. The file is located at `home > yuwantoro > Documents > configfile.go`. The code in the file is as follows:

```

1 package main
2
3 import (
4     "fmt"
5     "html"
6     "github.com/spf13/viper"
7     "net/http"
8 )
9
10 func main() {
11     // Set lokasi file config
12     viper.SetConfigFile("./config/env.json")
13
14     // Tampilkan error jika file config tidak ditemukan
15     if err := viper.ReadInConfig(); err != nil {
16         fmt.Println("Error reading config file, %s", err)
17     }
18 }

```

The terminal output shows the command `go run configfile.go` being executed. The output is:

```

[yuwantoro@localhost Documents]$ go run configfile.go
../go/src/github.com/spf13/viper/viper.go:43:2: no Go files in /home/yuwantoro/go/src/github.com/spf13/viper
../go/src/github.com/spf13/viper/util.go:21:2: cannot find package "github.com/spf13/afero" in any of:

```

Konfigurasi program dilakukan dengan menggunakan file `config.json` yang di dalamnya terdapat tipe data server dan atribut port. Dalam menggunakan viper, maka perlu dijabarkan data yang menjadi file konfigurasi. Path file konfigurasi dan nama file. Lalu file konfigurasi dibaca dengan sintaks viper. Fungsi http untuk mengambil port dari file `config.json` dengan menggunakan viper.