LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 1

MODUL 14

“KOMPOSISI”

DISUSUN OLEH:

RAIHAN ADI ARBA

103112400071

S1 IF-12-01

DOSEN:

Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI

Komposisi merupakan konsep fundamental dalam pemrograman berorientasi objek yang memungkinkan pembangunan tipe data kompleks melalui penggabungan tipe-tipe yang lebih sederhana. Konsep ini menjadi salah satu pilar penting dalam pengembangan perangkat lunak modern, memberikan alternatif yang lebih fleksibel dibandingkan dengan pendekatan pewarisan tradisional. Berbeda dengan pewarisan yang menerapkan hubungan "is-a" (adalah), komposisi mengedepankan hubungan "has-a" (memiliki), memberikan fleksibilitas lebih besar dalam pengembangan perangkat lunak dan mengurangi kompleksitas yang sering muncul dalam hierarki pewarisan.

Dalam bahasa pemrograman Go, komposisi diimplementasikan melalui mekanisme struct embedding, di mana satu struct dapat disisipkan ke dalam struct lain, memungkinkan sharing properti dan method antar struct. Pendekatan ini mencerminkan filosofi desain Go yang mengutamakan kesederhanaan dan kejelasan, sambil tetap mempertahankan kekuatan dan fleksibilitas yang diperlukan untuk pengembangan aplikasi modern. Struct embedding dalam Go memungkinkan pengembang untuk mencapai fungsionalitas yang serupa dengan pewarisan multiple, namun dengan cara yang lebih terkontrol dan mudah dipahami.

Pendekatan komposisi menawarkan berbagai keunggulan signifikan dalam pengembangan perangkat lunak. Pertama, modularitas yang lebih baik memungkinkan pengembang untuk memecah sistem kompleks menjadi komponen-komponen yang lebih kecil dan dapat dikelola. Kedua, maintainability yang lebih mudah karena perubahan pada satu komponen tidak selalu mempengaruhi komponen lain. Ketiga, reusability kode yang lebih tinggi karena komponen-komponen dapat digunakan kembali dalam berbagai konteks tanpa perlu memodifikasi kode yang ada.

Dalam konteks pengembangan aplikasi modern, komposisi menjadi semakin relevan karena kemampuannya untuk mendukung arsitektur yang lebih fleksibel dan adaptif. Pendekatan ini memungkinkan pengembang untuk dengan mudah menambah, mengurangi, atau memodifikasi fungsionalitas tanpa mempengaruhi struktur keseluruhan sistem. Implementasi komposisi dalam Go melalui struct embedding juga mendukung multiple embedding, memberikan developer lebih banyak pilihan dalam merancang arsitektur aplikasi mereka, sambil tetap mempertahankan prinsip-prinsip desain yang bersih dan mudah dipahami.

Dengan demikian, komposisi tidak hanya menjadi alternatif untuk pewarisan, tetapi juga merupakan pendekatan yang lebih sesuai untuk banyak skenario pengembangan modern. Kemampuannya untuk mendukung pengembangan yang modular, maintainable, dan reusable menjadikannya pilihan yang ideal untuk membangun sistem yang kompleks namun tetap mudah dikelola.

A. UNGUIDED

1. Latihan 1

Buatlah sebuah program Go yang digunakan untuk menghitung banyaknya bilangan ganjil dari 1 hingga n.

- Masukan: suatu bilangan bulat positif n
- Keluaran: teks yang menyatakan banyaknya bilangan ganjil yang terdapat antara 1 hingga n
- Catatan: Gunakan perulangan untuk pengecekan bilangan, bukan menggunakan operasi aritmatika

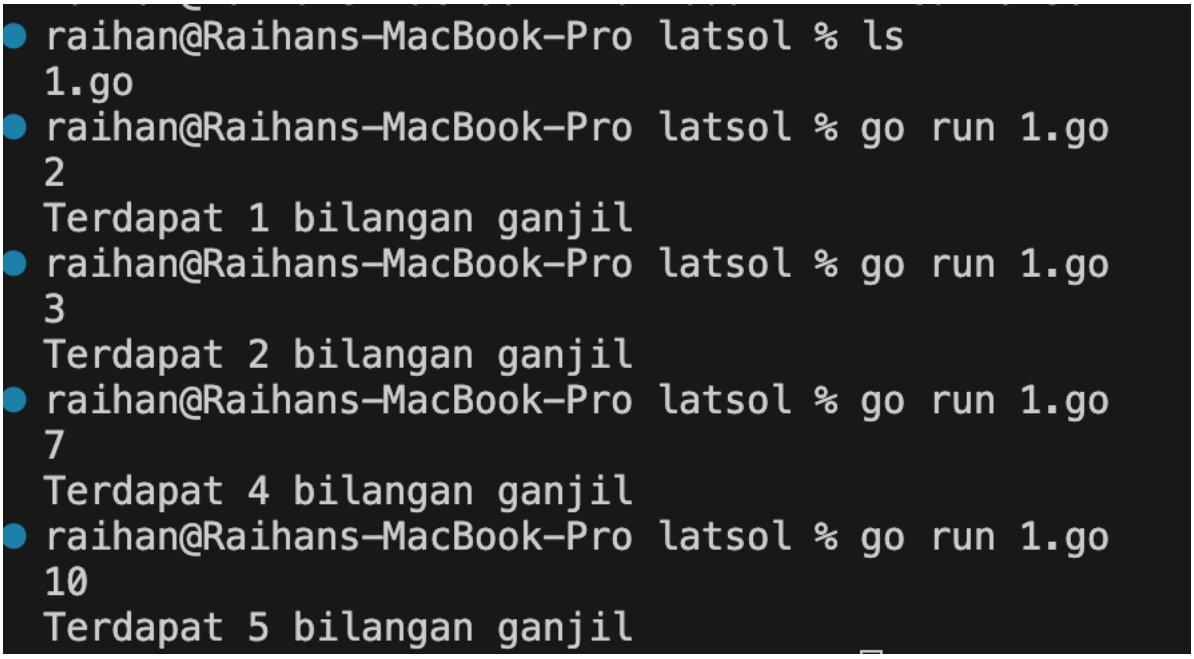
Contoh Masukan dan Keluaran:

No	Masukan	Keluaran
1	3	Terdapat 2 bilangan ganjil
2	2	Terdapat 1 bilangan ganjil
3	7	Terdapat 4 bilangan ganjil
4	10	Terdapt 5 bilangan ganjil

Source Code:

```
 package main

import "fmt"

func main() {
	var n, i, jmlganjil int

	fmt.Scan(&n)

	for i = 1; i <= n; i++ {
		if i%2 != 0 {
			jmlganjil++
		}
	}

	fmt.Printf("Terdapat %d bilangan ganjil\n", jmlganjil)
}
```

Output :



Penjelasan Program:

Program ini dirancang untuk menghitung jumlah bilangan ganjil dari 1 hingga n. Program dimulai dengan mendeklarasikan tiga variabel integer: n untuk menyimpan input batas atas, i sebagai variabel iterasi, dan jmlganjil untuk menghitung jumlah bilangan ganjil yang ditemukan. Program menggunakan fmt.Scan() untuk menerima input n dari pengguna. Selanjutnya, program menggunakan perulangan for yang berjalan dari 1 hingga n, dimana setiap iterasi akan memeriksa apakah bilangan tersebut ganjil dengan menggunakan operator modulo (i%2 != 0). Jika bilangan tersebut ganjil, maka variabel jmlganjil akan bertambah satu. Setelah perulangan selesai, program akan menampilkan total bilangan ganjil yang ditemukan dengan format "Terdapat X bilangan ganjil" dimana X adalah jumlah bilangan ganjil yang dihitung.

2. Latihan 2

Sebuah program digunakan untuk menentukan sebuah bilangan adalah prima atau bukan. Bilangan dikatakan prima apabila hanya memiliki faktor yaitu satu dan bilangan itu sendiri. Sebagai catatan bilangan satu bukanlah bilangan prima.

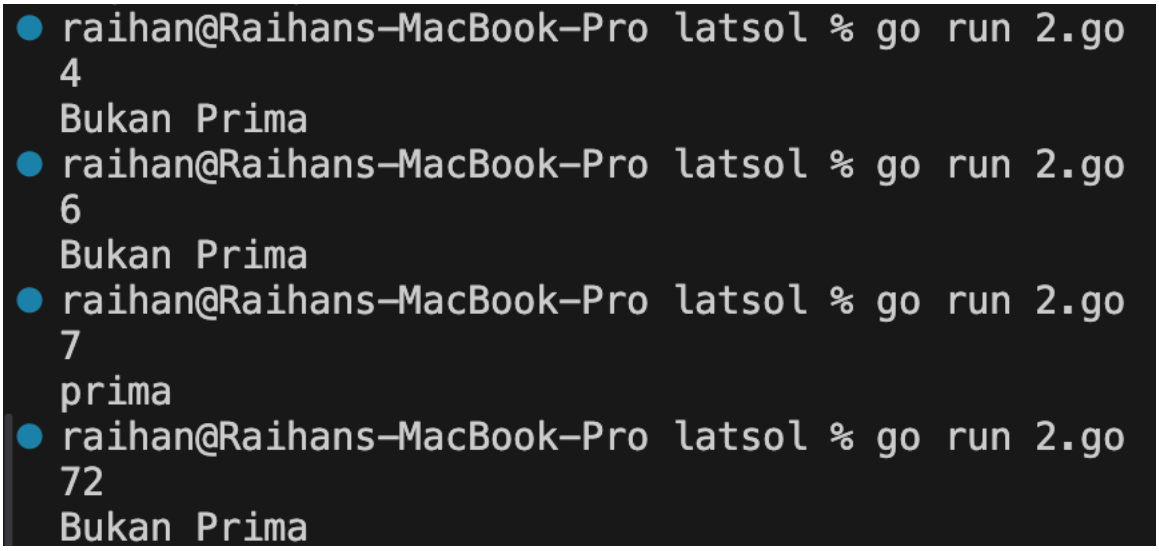
- Masukan: suatu bilangan bulat positif
- Keluaran: teks yang menyatakan bilangan adalah "prima" atau "bukan prima"

No	Masukan	Keluaran
1	5	prima
2	12	bukan prima
3	19	prima
4	72	bukan prima

Source Code :

```
package main

import "fmt"

func main() {
	var hasil string
	var n int

	fmt.Scan(&n)
	hasil = "prima"

	if n%n == 0 && n%2 != 0 {
		fmt.Println(hasil)
	} else {
		fmt.Println("Bukan Prima")
	}
}
```

Output :



Pembahasan :

Program ini mencoba untuk mengidentifikasi bilangan prima, namun memiliki logika yang kurang tepat. Program dimulai dengan mendeklarasikan variabel hasil bertipe string dan variabel n bertipe integer untuk menyimpan input bilangan. Program menggunakan fmt.Scan() untuk menerima input dari pengguna. Selanjutnya, variabel hasil diinisialisasi dengan nilai "prima". Program menggunakan kondisi if untuk memeriksa apakah bilangan tersebut prima dengan menggunakan dua kondisi: n%n == 0 (yang selalu bernilai 0 untuk semua bilangan) dan n%2 != 0 (yang hanya memeriksa apakah bilangan tersebut ganjil). Jika kedua kondisi terpenuhi, program akan mencetak "prima", jika tidak akan mencetak "Bukan Prima". Perlu diperhatikan bahwa logika pengecekan bilangan prima dalam program ini tidak tepat karena tidak memeriksa pembagian dengan bilangan-bilangan lain selain 2, dan tidak semua bilangan ganjil adalah bilangan prima.

3. Latihan 3

Siswa kelas IPA di salah satu sekolah menengah atas di Indonesia sedang mengadakan praktikum kimia. Di setiap percobaan akan menggunakan 4 tabung reaksi, yang mana susunan warna cairan di setiap tabung akan menentukan hasil percobaan. Siswa diminta untuk mencatat hasil percobaan tersebut. Percobaan dikatakan berhasil apabila susunan warna zat cair pada gelas 1 hingga gelas 4 secara berturutan adalah ‘merah’, ‘kuning’, ‘hijau’, dan ‘ungu’ selama 5 kali percobaan berulang.

Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan true apabila urutan warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan false untuk urutan warna lainnya.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

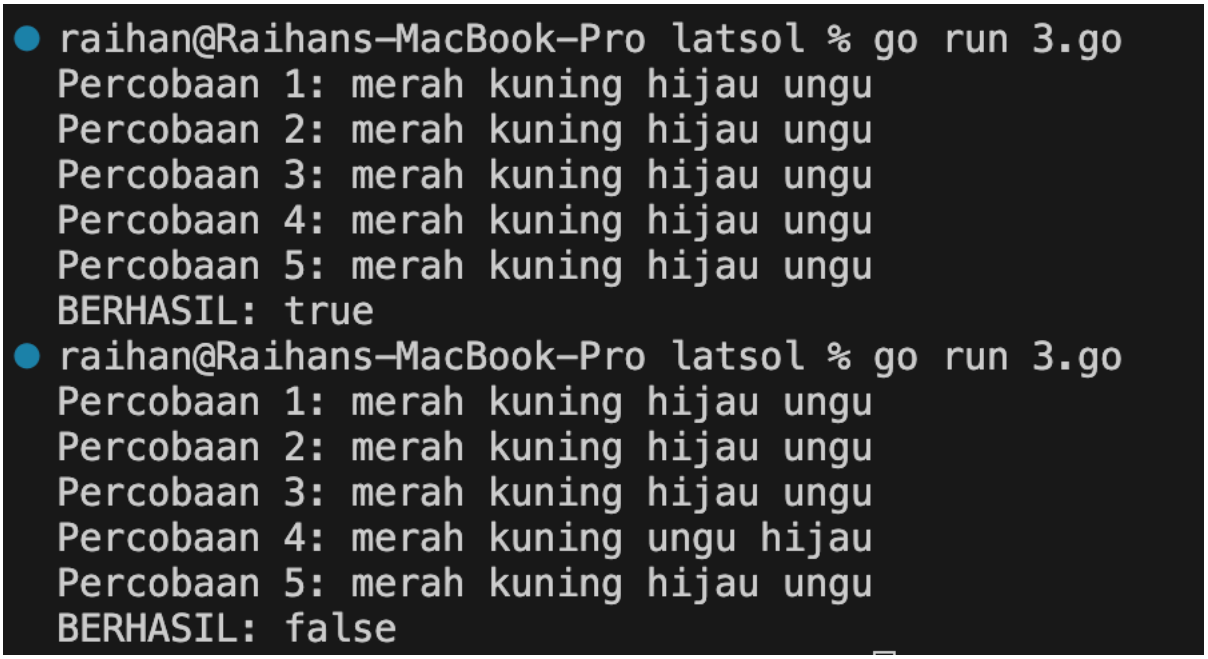
Contoh masukan dan keluaran:

```
Percobaan 1: merah   kuning   hijau   ungu
Percobaan 2: merah   kuning   hijau   ungu
Percobaan 3: merah   kuning   hijau   ungu
Percobaan 4: merah   kuning   hijau   ungu
Percobaan 5: merah   kuning   hijau   ungu
BERHASIL: true

Percobaan 1: merah   kuning   hijau   ungu
Percobaan 2: merah   kuning   hijau   ungu
Percobaan 3: merah   kuning   hijau   ungu
Percobaan 4: ungu    kuning   hijau   merah
Percobaan 5: merah   kuning   hijau   ungu
BERHASIL: false
```

Source Code:

```
package main

import "fmt"

func main() {
	var warna1, warna2, warna3, warna4 string
	berhasil := true

	for i := 1; i <= 5; i++ {
		fmt.Printf("Percobaan %d: ", i)
		fmt.Scan(&warna1, &warna2, &warna3, &warna4)

		if warna1 != "merah" || warna2 != "kuning" ||
			warna3 != "hijau" || warna4 != "ungu" {
			berhasil = false
		}
	}

	fmt.Printf("BERHASIL: %t\n", berhasil)
}
```

Output:



Deskripsi Program:

Program ini merupakan sistem pengecekan urutan warna yang terdiri dari 5 kali percobaan. Program dimulai dengan mendeklarasikan empat variabel string (warna1, warna2, warna3, warna4) untuk menyimpan input warna, dan variabel boolean berhasil yang diinisialisasi dengan nilai true. Program menggunakan perulangan for yang akan berjalan sebanyak 5 kali, dimana setiap iterasi meminta input empat warna secara berurutan. Pada setiap percobaan, program akan memeriksa apakah urutan warna yang diinput sesuai dengan urutan yang diharapkan yaitu "merah", "kuning", "hijau", dan "ungu". Jika dalam satu percobaan terdapat urutan warna yang tidak sesuai, variabel berhasil akan diubah menjadi false. Setelah semua percobaan selesai, program akan menampilkan hasil akhir berupa status keberhasilan (true/false) yang menunjukkan apakah semua percobaan memiliki urutan warna yang tepat atau tidak.

4. Latihan 4

Buatlah sebuah program Go yang digunakan untuk menghitung banyaknya bilangan ganjil dari 1 hingga n.Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan ‘–‘, contoh pita diilustrasikan seperti berikut ini.

Pita: mawar – melati – tulip – teratai – kamboja – anggrek

Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita. (Petunjuk: gunakan operasi penggabungan string dengan operator “+”).

Tampilkan isi pita setelah proses input selesai.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

N: **3** Bunga 1: **Kertas** Bunga 2: **Mawar** Bunga 3: **Tulip** Pita: Kertas – Mawar – Tulip -	N : **0** Pita :

Modifikasi program sebelumnya, proses input akan berhenti apabila user mengetikkan ‘SELESAI’. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

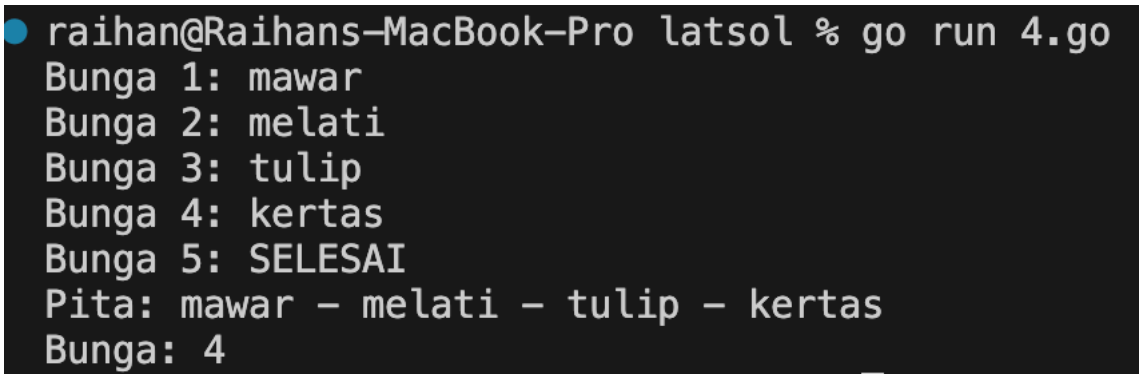
Bunga 1: **Kertas** Bunga 2: **Mawar** Bunga 3: **Tulip** Bunga 4: **SELESAI** Pita: Kertas – Mawar – Tulip - Bunga: 3	Bunga 1: **SELESAI** Pita : Bunga: 0

Source Code:

```
package main

import "fmt"

func main() {
	var bunga, pita string
	var jumlahBunga int

	for {
		fmt.Printf("Bunga %d: ", jumlahBunga+1)
		fmt.Scan(&bunga)

		if bunga == "SELESAI" {
			break
		}

		if jumlahBunga == 0 {
			pita = bunga
		} else {
			pita = pita + " - " + bunga
```

```
            }
            jumlahBunga++
        }

        fmt.Printf("Pita: %s\n", pita)
        fmt.Printf("Bunga: %d\n", jumlahBunga)
}
```

Output :



Penjelasan Program:

Program ini merupakan sistem input bunga yang membentuk sebuah pita. Program dimulai dengan mendeklarasikan variabel bunga dan pita sebagai string untuk menyimpan nama bunga dan susunan pita, serta variabel jumlahBunga sebagai integer untuk menghitung total bunga yang diinput. Program menggunakan perulangan tanpa batas yang akan terus meminta input nama bunga dari pengguna dengan format "Bunga X:" dimana X adalah nomor urut bunga. Perulangan akan berhenti ketika pengguna menginput kata "SELESAI". Dalam proses pembentukan pita, jika input adalah bunga pertama, nama bunga langsung disimpan ke variabel pita, sedangkan untuk bunga-bunga selanjutnya, nama bunga akan ditambahkan ke pita dengan dihubungkan menggunakan tanda " - ". Setiap kali input berhasil, counter jumlahBunga akan bertambah. Setelah perulangan selesai, program akan menampilkan susunan pita final dengan format "Pita: [nama-nama bunga]" dan total jumlah bunga yang telah diinput. Program ini efektif untuk membuat rangkaian atau pita bunga dengan format yang rapi dan terstruktur.

DAFTAR PUSTAKA

Prayogo, N. A. (2021). ***Dasar Pemrograman Go. Ebook***