MODUL PENDAHULUAN KURSUS GOLANG FOR INTERMEDIATE

Versi	1.0
Tahun Penyusunan	2019

Laboratorium Pengembangan Komputerisasi

UNIVERSITAS GUNADARMA

KATA PENGANTAR

Modul ini merupakan modul pendahuluan yang disusun sebagai materi persiapan bagi mahasiswa peserta kursus sebelum setiap pertemuan kursus yang akan dimulai. Materi setiap pertemuan selalu dimulai dengan penjelasan tentang objektif yang akan dicapai dari proses belajar dalam pertemuan tersebut.
Modul GOLANG terdiri dari 8 pertemuan materi :

1. Test Driven Development dan Unit Test
2. Function & Method
3. Structure Query Language (SQL) pada Bahasa Pemrograman Go
4. HTTP REQUEST Pada Bahasa Pemrograman GO
5. Polymer : Properties, Data Binding, Observer & Compute
6. Polymer: Layout Styling 1
7. Polymer: Layout Styling 2
8. Deployment

Secara umum, materi pendahuluan pada setiap pertemuan akan menjelaskan konsep atau teori mengenai topik yang akan dibahas, dan menerangkan secara garis besar langkah yang diperlukan dalam menjalankan suatu software utilitas atau aplikasi yang mendukung pembahasan topik yang dibahas dalam materi tersebut.
Setiap peserta kursus sangat diharapkan untuk mempelajari dengan seksama modul pendahuluan ini, mengingat pemahaman yang baik atas materi ini, akan sangat membantu pada waktu proses belajar selama kegiatan kursus berlangsung, terutama dalam mengerjakan soal-soal Pre-test.
Untuk mengikuti perkembangan teknologi di masa mendatang, maka modul pendahuluan ini disusun oleh Lembaga Pengembangan Komputerisasi Universitas Gunadarma dengan bantuan tim yang bekerja secara penuh yaitu, Drs. Hasin Widjadi, MMSI , Ricky Agus T., ST., SSi., MM , Rheza Andika., SKom., MMSI. , Guntur Eka Saputra, ST., MMSI. , Ahmad Hidayat, SKom., MMSI. , Octaviani Hutapea, ST. , Aria Samudera Elhamidy, SKom. , Azman Agung Nugraha , Ilham Muhammad.

Jakarta, Oktober 2019

Lembaga Pengembangan Komputerisasi

Universitas Gunadarma

DAFTAR ISI

1 Test Driven Development dan Unit Test

Objektif :

- **Mahasiswa diharapkan mampu memahami konsep Test Driven Development**
- **Mahasiswa diharapkan dapat menerapkan konsep Test Driven Development dan Unit Test dalam pembuatan program**

1.1 Pendahuluan

Mengapa sebuah program harus melewati proses pengujian? Terdapat sebuah studi yang dilakukan oleh National Institute of Standart and Technologi (NIST) pada tahun 2002 yang berisikan bahwa bugs pada perangkat lunak menyebabkan kerugian ekonomi di Amerika Serikat sebesar $59.5 billion tiap tahunnya pada saat itu, menurut penelitian tersebut sepertiga dari kerugian ini bisa dihindari jika dilakukan software testing yang lebih baik.

Tujuan utama dari pengujian perangkat lunak sebenarnya sederhana yaitu untuk memastikan bahwa software yang dihasilkan sesuai dengan kebutuhan (requirement) yang sebelumnya ditentukan.

Di bawah ini merupakan beberapa alasan mengapa sebuah program harus melalui tahap pengujian:

1. Untuk menemukan apakah masih terdapat kesalahan atau kekurangan pada program
2. Untuk menyempurnakan program yang baru dibuat jika masih ada kekurangan
3. Untuk membuktikan apakah program tersebut sudah sesuai outputnya seperti yang diinginkan user
4. Untuk melihat performa dari program tersebut
5. Dan hal lain yang menjadi penyebab diperlukanya pengujian pada program

1.2 Unit Test

Unit testing adalah metode verifikasi dan validasi dimana programmer melakukan pengujian terhadap setiap unit pada source code. Sebuah unit merupakan bagian terkecil dari aplikasi yang dapat di uji coba. Unit adalah fungsi individual atau prosedur. Unit test

biasanya dibuat oleh developer dan akan berbeda script dengan tester sendiri. dimana tester memiliki sebuah test sendiri yang biasanya menangani test tentang Functional end point seperti interaksi pengguna dengan aplikasi, API testing, dan lain-lain. Pada Bahasa pemrograman Go disediakan package testing, yang berisikan banyak tools yang dapat digunakan untuk keperluan unit testing.

- TESTING

  Pertama yang harus dibuat merupakan file yang akan di test conoth disimpan dengan nama file latihan.go contoh file berisikan perhitungan luas dan keliling Persegi sederhana.

```
package main

import "math"

type Persegi struct {
    Sisi float64
}

func (k Persegi) Luas() float64 {
    return k.Sisi * k.Sisi
}

func (k Persegi) Keliling() float64 {
    return k.Sisi * 4
}
```

File untuk keperluan testing dipisah dengan file utama, namanya harus berakhiran _test.go, ***namafileutama_test.go*** dan package-nya harus sama. Pada bab ini, file utama adalah latihan.go, maka file testing harus bernama latihan_test.go.

Unit test di Golang dituliskan dalam bentuk fungsi, yang memiliki parameter yang bertipe *testing.T, dengan nama fungsi harus diawali kata Test (pastikan sudah meng-import package testing sebelumnya). Dengan menggunakan parameter tersebut, bisa mengakses method-method untuk keperluan testing. Pada contoh di bawah ini disiapkan 3 buah fungsi test, yang masing-masing digunakan untuk mengecek apakah hasil kalkulasi luas dan keliling Persegi adalah benar.

```
package main
import "testing"
var (
    Persegi              Persegi   = Persegi{5}
    luasSeharusnya     float64 = 25
    kelilingSeharusnya float64 = 20
)
```

```
func TestHitungLuas(t *testing.T) {
    t.Logf("Luas : %.2f", Persegi.Luas())
    if Persegi.Luas() != luasSeharusnya {
        t.Errorf("SALAH! harusnya %.2f", luasSeharusnya)
    }
}
func TestHitungKeliling(t *testing.T) {
    t.Logf("Keliling : %.2f", Persegi.Keliling())
    if Persegi.Keliling() != kelilingSeharusnya {
        t.Errorf("SALAH! harusnya %.2f", kelilingSeharusnya)
    }
}
```

- **Eksekusi File Testing**

  Proses eksekusi file testing adalah menggunakan command go test. Karena struct yang diuji berada dalam file **latihan.go**, maka pada saat eksekusi test menggunakan go test, nama file **latihan_test.go** dan **latihan.go** perlu dituliskan. Berikut merupakan penulisannya:

```
go test namafileutama.go namafileutama_test.go -v
```

  Contoh:

```
go test latihan.go latihan_test.go -v
```

  -v atau –verbose merupakan digunakan untuk menampilkan seluruh output log pada saat user melakukan pengujian.

- ***Method Testing***

  Berikut merupakan tabel dari method testing:

Method	Kegunaan
Log()	Menampilkan log
Logf()	Menampilkan log menggunakan format
Fail()	Menandakan terjadi Fail() dan proses testing fungsi tetap diteruskan
FailNow()	Menandakan terjadi Fail() dan proses testing fungsi dihentikan
Failed()	Menampilkan laporan fail
Error()	Log() diikuti dengan Fail()
Errorf()	Logf() diikuti dengan Fail()
Fatal()	Log() diikuti dengan failNow()
Fatalf()	Logf() diikuti dengan failNow()
Skip()	Log() diikuti dengan SkipNow()

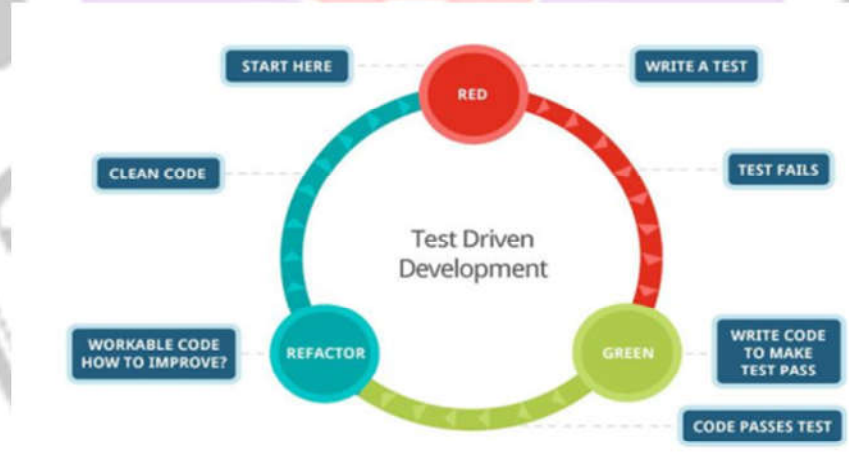
Skipf()	Logf() diikuti dengan SkipNow()
SkipNow()	Menghentikan proses testing fungsi, dilanjutkan ke testing fungsi setelahnya
Skiped()	Menampilkan laporan skip
Parallel()	Menge-set bahwa eksekusi testing adalah parallel

1.3 *Test Driven Development*

- **Definisi**

  Test Driven Development (TDD) adalah teknik pengembangan perangkat lunak yang di mana langkah pengerjaanya adalah membuat kode uji terlebih dahulu sebelum kode produksi dan *refactoring*. Metode TDD mengharuskan pengembang untuk menentukan terlebih dahulu kebutuhan dan desain dari perangkat lunak yang akan dibuat baru kemudian melakukan implementasi kode. Satu poin penting dalam metode TDD adalah pengembangan perangkat lunak fokus pada spesifikasi bukan pada validasinya.

- **Konsep**



Gambar 1.1 Konsep TDD

Pada gambar 1.1 di atas dapat dijelaskan bahwa proses dalam melakukan pengujian terhadap suatu program oleh pengembang menggunakan metode TDD pada awalnya adalah:

1. Sebelum menuliskan kode program, maka tuliskan *test-* nya terlebih dahulu. Pengembang harus memikirkan semua kemungkinan yang dapat terjadi untuk input dan output dari program yang akan dibuat.
2. Jalankan *test-code* tersebuit, dan pastikan *test-code* nya *fail* karena belum ada kode apapun untuk membuat *test-* nya *pass* (sukses) .
3. Ketik *working code* seminimum mungkin dengan tujuan agar *test-* nya *pass*
4. Jalankan *test* dan cek apakah *test-* nya *pass* . Jika belum *pass* , maka perbaiki *working code* sampai memenuhi ekspektasi dari *test* .
5. Jika *working code* yang ditulis sebelumnya menjadi tidak sesuai posisinya. TDD menyediakan fungsi *Refactor the code* , untuk merapihkan kembali kode yang dibuat. Selama *test-* yang dibuat masih *pass,* memiliki arti bahwa kode yang dibuat tidak ada masalah dengan kode yang di *refactor* tersebut.
6. Proses dari no 1–5 dapat diulang untuk fungsi-fungsi lainnya .

- **Kelebihan**

  Berikut merupakan kelebihan dari metode *Test Driven Development*

  1. Pengecekan error lebih mudah dilakukan
  2. Simulasi manual untuk pengujian fungsi perangkat lunak menjadi berkurang
  3. Dengan begitu pengembang dapat melakukan pengcekan lebih cepat
  4. Memudahkan pengguna dalam memodifikasi perubahan
  5. Memudahkan pengembang dalam menemukan *bug*

- **Kekurangan**

  TDD adalah pendekatan yang baik untuk pengembangan perangkat lunak baru tetapi lebih memakan waktu proses model ketika menguji sistem perangkat lunak yang ada.