Lab 4
Class, Method dan Access Specifier

Dasar-Dasar Pemrograman 2
CSGE601021
Semester Genap 2016/2017

Batas waktu pengumpulan:
Senin, 6 Maret 2017 pukul 18.00 Waktu Scele (Lab Senin)

Tujuan dari Lab ini adalah melatih Anda agar menguasai bahan kuliah yang diajarkan di kelas. Mahasiswa diperbolehkan untuk berdiskusi, tetapi Anda tetap harus **menuliskan sendiri** solusi/kode program dari soal yang diberikan tanpa bantuan orang lain. Belajarlah menjadi mahasiswa yang mematuhi integritas akademik. **Sikap Jujur merupakan sebuah sikap yang dimiliki mahasiswa Fasilkom UI.**

Peringatan: Jangan mengumpulkan pekerjaan beberapa menit menjelang batas waktu pengumpulan karena ada kemungkinan pengumpulan gagal dilakukan atau koneksi internet terputus!

Lab 4
Class, Method dan Access Specifier

Object Oriented Programming

Dalam paradigma Object Oriented Programming, kita memodelkan objek dunia nyata sebagai sebuah class di dalam program. Objek tersebut memiliki variabel, constructor, dan method. Sebagai contoh, mari kita lihat sebuah mobil. Mobil dapat dilihat sebagai sebuah objek, yang memiliki *merk*, *model,* dan *warna*. Saat bergerak, mobil memiliki kelajuan. Mobil dapat digas (akselerasi) dan direm (deselerasi). Oleh karena itu, secara simpel kita dapat memodelkan sebuah mobil sebagai sebuah kelas:

```
public class Mobil {
    private String merk;
    private String model;
    private String warna;
    private double kelajuan;

    public Mobil(String merk, String model, String warna) {
        this.merk  = merk;
        this.model = model;
        this.warna = warna;
        kelajuan = 0;
    }

    public void akselerasi() { //akselerasi sebesar 10 km/h
        kelajuan += perubahan;
    }

    public void deselerasi() { //deselerasi sebesar 10 km/h
        if (kelajuan > 10) {
            kelajuan -= 10};
        } else {
            kelajuan = 0;
        }
    }
}
```

Dapat dilihat di contoh di atas bahwa mobil objek mobil memiliki instance variable (merk, model, warna, kelajuan), constructor, dan method akselerasi serta deselerasi.

Method

Terdapat empat jenis method yang dapat ditambahkan ke objek:

- Setter: Mengubah instance variable
- Getter: Mengambil nilai instance variable
- toString: Sebuah method yang mengatur bagaimana class tersebut akan dituliskan dalam bentuk String. Secara default, apabila kita melakukan perintah print pada sebuah objek, yang akan dicetak adalah nama class-nya serta alamat objek tersebut. Kita dapat mengambil-alih/*override* fungsi ini sehingga ketika objek tersebut di-print, yang ditulis sesuai dengan keinginan kita. Sebagai contoh, kita dapat membuat fungsi toString untuk class Mobil di atas:

```
@Override
public String toString() {
    return "Mobil " + merk + " " + model + " warna " + warna;
}
```

  Sehingga apabila di-cetak, dapat dihasilkan output semisal "Mobil Tayoto Rycam warna hitam".
- Method-method lain

Encapsulation

Encapsulation merupakan proses menutup/membungkus instance variable. Dalam proses encapsulation, instance variable yang dimiliki kelas diberikan access type *private*. Untuk memungkinkan akses variabel tersebut dari kelas lain, dapat disediakan method-method dengan access type *public* yang dapat memanipulasi variabel tersebut. Apabila diperlukan, dapat diberikan setter dan getter dengan access type *public.* Tujuan encapsulation adalah akses terhadap variabel class tersebut terkontrol, sehingga lebih aman. Sebagai contoh, apabila class Mobil di atas tidak di-*encapsulate*, seorang pengemudi dapat melakukan hal berikut:

```
// Buat mobil baru, kecepatan 0
Mobil mobilku = new Mobil ("Bishimitsu", "Penombak", "Merah");

// Langsung di-turbo
```

```
mobilku.kecepatan = 1000

// Direm mendadak
mobilku.kecepatan = 0
```

Agar *behavior* class yang kita kembangkan dapat sesuai dengan harapan, dan agar pengguna class tersebut dapat menggunakannya dengan lebih mudah, adalah alasan mengapa *encapsulation* penting untuk dilakukan.

Access Type

Access type menentukan bagian program mana saja yang dapat mengakses suatu class, variabel, maupun method. Ada empat access type di Java:

- Private : Hanya dapat diakses oleh class yang memilikinya
- Default : Dapat diakses dari semua class di dalam package
- Protected : Dapat diakses dari semua class di dalam package, serta subclass/anak kelas dari class tersebut
- Public : Dapat diakses dari semua class

Static

Static adalah sebuah label yang dapat diberikan kepada variabel dalam kelas. Sebelumnya, instance variable tiap objek berbeda satu sama lain; dan apabila variabel suatu instance diubah, instance lain tidak ikut berubah. Namun, apabila sebuah variabel dibuat menjadi static, maka variabel tersebut akan menjadi "sama" untuk setiap instance, dan setiap instance objek tersebut akan memiliki nilai variabel yang sama. Apabila nilai variabel static tersebut diubah di satu instance diubah, nilai variabel tersebut juga ikut berubah apabila diakses dari instance-instance lain.

Design Class

Untuk mendesign class yang baik ada beberapa hal yang harus diperhatikan, seperti:

1. Menentukan aktor dari class tersebut, maksudnya adalah menentukan objek yang nantinya dapat melakukan sesuatu atau menjadi "korban" dengan kata lain sebuah aksi tidak dapat dijadikan suatu class. Contoh : Buatlah sebuah program yang menyelesaikan permasalahan permainan catur, salah satu aktor dari problem ini adalah pemain, oleh karena itu nantinya dalam solusi dipastikan terdapat satu class

yang menggambarkan seorang pemain, sebut saja class Player tetapi kita tidak akan membuat class MenggerakanBidak.

2. Walaupun tidak punya aktor, class juga sebenarnya tetap dapat dibuat, untuk kebutuhan-kebutuhan lain (utility function). Contohnya : class Math.
3. Perlu tidaknya memiliki method main (program utama).
4. Cohesion dan Coupling.

Cohesion

Cohesion adalah tingkat ke-"fokus"-an sebuah class; atau seberapa fokus sebuah kelas merepresentasikan sebuah benda/konsep. Keuntungan dari pengembangan dengan cohesion yang tinggi adalah kemudahan bagi kita maupun pengguna untuk memahami class tersebut. Selain itu, dalam mengembangkan program kita menginginkan class-class yang spesifik untuk tujuan kita saja, dan tambahan-tambahan lainnya akan tidak berguna. Untuk mencapai cohesion yang tinggi, semua method maupun variabel di dalam class seharusnya merepresentasikan hanya sebuah konsep, dan tidak yang lain.

Sebagai contoh, kelas Mobil yang di atas seharusnya hanya mengandung keperluan dari sebuah mobil saja. Apabila kita ingin menambahkan fitur untuk membeli mobil, jangan ditambahkan di dalam kelas Mobil, melainkan di sebuah kelas baru yang bertanggungjawab untuk pembelian saja.

Coupling

Coupling adalah tingkat keberkaitan suatu class kepada class-class yang lain. Class A dianggap berkaitan dengan class B jika:

- B disimpan sebagai variabel di A
- B merupakan parameter method di A
- B merupakan return type method di A

Ketika class A berkaitan dengan class B, ada beberapa konsekuensi:

- Ketika kita membutuhkan class A, kita harus mengikutkan class B
- Ketika kita memodifikasi class B, kita mungkin harus memodifikasi class A pula

Desain yang baik adalah ketika tingkat coupling keseluruhan di program rendah.

Sebagai contoh, apabila kita ingin menambahkan class Roda di program di atas, dan menyimpan Roda di dalam Mobil, maka Mobil akan berkaitan dengan Roda. Seharusnya, class Roda cukup menyimpan informasi tentang roda itu sendiri, dan tidak perlu berkaitan dengan Mobil pula.

Jadi kesimpulannya, sebuah software / program memiliki design yang baik apabila high cohesion (fokus pada tujuan dari kelas tersebut saja) dan low coupling (independent terhadap kelas lain).

Soal Tutorial

Menggunakan program interview yang telah dikembangkan sebelumnya, Agung, sang karyawan HRD perusahaan iSUS berhasil merekrut sejumlah karyawan baru di perusahaannya. Namun, Agung sekarang menghadapi masalah baru. Banyaknya karyawan baru di perusahaan iSUS membuat Agung kesulitan dalam melacak tiap karyawan serta pekerjaan yang di-*assign* pada mereka! Karena bobot pekerjaan yang dilakukan tiap karyawan berpengaruh pada jumlah bonus tunjangan kepada karyawan, pelacakan ini perlu dilakukan dengan seksama. Karena merasa sangat terbantu oleh program interview sebelumnya, Agung memiliki ide untuk mengembangkan sebuah program sederhana baru untuk melacak tiap karyawan, tiap task yang ada, dan karyawan mana yang mengerjakan tiap task. Bantulah Agung menciptakan program tersebut!

Pada program telah terdapat kelas Task, Employee, dan kelas utama. Berikut penjelasan singkat tiap kelas.

Task.java
Kelas yang merepresentasikan suatu task.

- name : nama tugas
- bobot : bobot pekerjaan sebuah tugas
- selesai : status tugas (sudah selesai atau belum)

Employee.java
Kelas yang merepresentasikan seorang karyawan.

- name : nama karyawan
- employeeId : id karyawan
- pangkat : pangkat karyawan
  - **1**: Newbie
  - **2**: Junior
  - **3**: Senior
- divisi : divisi karyawan
  - **0**: Engineering
  - **1**: Data
  - **2**: Marketing
- totalBobot : total bobot tugas yang telah dikerjakan
- gaji : gaji karyawan (rumus: pangkat * 5000000)
- temanTerbaru : nama teman (karyawan) terbaru di perusahaan

workOnTask	Method menerima sebuah Task/tugas sebagai parameter. Apabila status tugas belum selesai (selesai == false), tandai sebagai selesai dan tambahkan bobot tugas (bobot) ke total bobot pekerjaan (totalBobot) milik karyawan. Lalu, cetak pernyataan dengan format: "[nama karyawan] telah mengerjakan tugas [nama tugas] dengan bobot [bobot tugas]" Apabila status tugas sudah selesai (selesai == true), cetak pernyataan:

	"[nama karyawan] gagal mengerjakan tugas [nama tugas] karena status sudah selesai"
calculateBonus	Menghitung nilai uang bonus untuk karyawan. Dihitung dari: ● Newbie : (totalBobot - 5) * 1000000 ● Junior : (totalBobot - 10) * 1000000 ● Senior : (totalBobot - 15) * 1000000 Nilai bonus tidak mungkin minus. Dengan kata lain, apabila hasil bonus < 0, nilai bonus yang didapatkan adalah 0.
dispenseWages	Mencetak jumlah uang yang dibayarkan ke karyawan. Formatnya adalah: "[nama karyawan] telah menerima gaji sebesar [gaji] dengan bonus sebesar [jumlah bonus]"

Dengan kelas-kelas yang ada, lakukan tugas-tugas berikut:

1. Lakukan *encapsulation* kepada *instance variable* yang ada agar mengikuti *best practice* pemrograman Java.
2. Implementasi method-method yang ada di kelas Employee sesuai dengan instruksi di atas.
3. Lakukan *override* terhadap method toString() untuk Employee dan Task dengan format sebagai berikut:
    - Employee: "Karyawan #[id karyawan]: [nama karyawan], [divisi] [pangkat]"
    - Task: "Task: [nama task], bobot: [bobot task], status: [Selesai|Belum selesai]"
4. Lihatlah kembali kelas Employee. Dapat dilihat bahwa beberapa *instance variable* maupun method yang tidak sesuai pada tempatnya, dan menyebabkan *low cohesion*. Buatlah sebuah kelas baru dan pindahkanlah beberapa method ke kelas tersebut sehingga tercipta *high cohesion*! (Hint: method-method dan variabel yang berkaitan dengan gaji karyawan)
5. Setelah menyelesaikan soal no. 4, ubah method main agar bisa mencetak jumlah uang yang dibayarkan (dispenseWages).
6. Pelajari perilaku variabel static temanTerbaru, dengan mencetak nilainya tiap kali sebuah karyawan diciptakan. Amati bahwa isi dari variabel tersebut milik semua karyawan yang ada akan selalu berubah tiap kali karyawan baru ditambahkan.

Sebagai contoh, dengan template input seperti pada method main, program akan menghasilkan:

```
Karyawan #1SE40: Ann, Senior Engineering
Karyawan #1SE25: Bob, Junior Data
Karyawan #1PM21: Charlie, Newbie Marketing
Task: Implement first feature, bobot: 10, status: Belum selesai
Task: Implement second feature, bobot: 10, status: Belum selesai
Task: Supervise development, bobot: 20, status: Belum selesai
Task: Implement first feature, bobot: 10, status: Selesai
Task: Implement second feature, bobot: 10, status: Selesai
```

```
Task: Supervise development, bobot: 20, status: Selesai
Ann telah menerima gaji sebesar 15000000 dengan bonus sebesar 0
Bob telah menerima gaji sebesar 10000000 dengan bonus sebesar 5000000
Charlie telah menerima gaji sebesar 5000000 dengan bonus sebesar 15000000
```

Format Pengumpulan

- KELAS_NPM_TUTORIAL4.zip contoh: B_1606123456_TUTORIAL4.zip
- Isi file KELAS_NPM_TUTORIAL4.zip:
  - DDP2Tutorial4.java (kelas utama yang digunakan untuk menjalankan program)
  - Employee.java
  - Task.java
  - Kelas tambahan seperlunya