Lab 4
Class, Method dan Access Specifier

Dasar-Dasar Pemrograman 2
CSGE601021
Semester Genap 2016/2017

Batas waktu pengumpulan:
Rabu, 8 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

Jack baru saja mewarisi sebuah ladang pertanian dari kakeknya yang baru meninggal dunia. Jack dengan bersegera pindah ke ladang tersebut untuk mulai bercocoktanam. Dengan bersemangat ia pun menanam begitu banyak pohon buah. Saking banyaknya pohon yang ia tanam, ketika tiba saatnya untuk panen buah, Jack pun kewalahan untuk memanen semua buah yang ada! Ia harus panen semua buah yang ada pada hari itu juga, atau buah-buah yang ada akan busuk atau dicuri oleh hewan-hewan liar. Untungnya, beberapa hari sebelum hari panen, Jack menemukan sebuah pohon berisi peri-peri pembantu, yang biasa dikenal dengan Harvest Sprites. Peri-peri ini siap membantu Jack untuk memanen buah yang ada, namun dengan sebuah syarat! Mereka hanya akan membantu panen jika Jack membayar mereka dengan buah berry, serta jamur-jamur langka bernama Truffle. Jumlah berry yang diberikan berbanding lurus dengan total harga buah yang dipanen. Jumlah Truffle yang diminta tiap peri pun berbeda, tergantung oleh tingkat kecintaan peri tersebut kepada Jack. Jack pun kewalahan untuk menghitung jumlah Truffle yang dibayarkan untuk masing-masing peri.

Untungnya, Jack yang merupakan seorang alumni Fakultas Ilmu Komputer Universitas Indonesia menyadari bahwa masalah ini bisa dipecahkan menggunakan Object-Oriented Programming. Jack telah menuliskan sedikit kode untuk memecahkan masalah ini, namun masih belum selesai. Bantulah Jack menyelesaikan program tersebut.

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

Fruit.java
Kelas yang merepresentasikan suatu buah.

- nama : nama buah
- harga : harga buah
- sudahPanen : status buah (sudah panen atau belum)

Sprite.java
Kelas yang merepresentasikan seorang peri.

- nama : nama peri
- heart : jumlah 'hati', tingkat kecintaan peri ke sang petani
- totalHarga : total bobot tugas yang telah dikerjakan
- periTerakhir : nama peri yang terakhir datang ke kebun

harvest	Method menerima sebuah Fruit/buah sebagai parameter. Apabila status buah belum dipanen (sudahPanen == false), tandai sudah panen dan tambahkan harga buah ke total harga panen buah (totalBobot) milik peri. Lalu, cetak pernyataan dengan format: "[nama peri] telah memanen buah [nama buah] dengan harga [harga buah] G" Apabila status buah sudah dipanen (sudahPanen == true), cetak pernyataan:

	"[nama peri] tidak jadi memanen buah [nama buah] karena sudah tidak ada"
calculateTruffles	Menghitung jumlah truffle yang diberikan kepada peri. Dihitung dengan: ● Jumlah heart < 5 (5 - heart) ● Jumlah heart >= 5: (10 - heart) * 2 Jumlah truffle yang diberikan tidak mungkin minus. Dengan kata lain, apabila jumlahnya < 0, jumlah truffle yang diberikan adalah 0.
calculateBerries	Menghitung jumlah berry yang diberikan kepada peri. Dihitung dengan: ● Total harga panen < 100 G: 10 ● Total harga panen >= 100 G: totalHarga - 90 Jumlah berry yang diberikan tidak mungkin minus. Dengan kata lain, apabila jumlahnya < 0, jumlah berry yang diberikan adalah 0.
paySprite	Mencetak jumlah truffle yang diberikan ke peri. Formatnya adalah: "Atas jasa [nama peri] dalam memanen sejumlah [total harga panen peri] G, [nama peri] telah menerima [jumlah berry] buah berry dan [jumlah truffle] buah truffle" Namun, apabila sang peri tidak memanen buah apapun, cukup cetak keluaran dengan format: "Karena [nama peri] tidak memanen satu pun buah, [nama peri] tidak diberikan bayaran apapun"

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 Sprite sesuai dengan instruksi di atas.
3. Lakukan *override* terhadap method toString() untuk Sprite dan Fruit dengan format sebagai berikut:
    - Sprite: "Peri [nama peri], jumlah hati: [jumlah hari peri]
    - Fruit: "Buah [nama buah], harga: [harga buah] G, status: [Sudah|Belum dipanen]"
4. Lihatlah kembali kelas Sprite. Dapat dilihat bahwa ada beberapa 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 menghitung dan memberikan truffles)
5. Setelah menyelesaikan soal no. 4, ubah method main agar bisa mencetak jumlah uang yang dibayarkan (dispenseWages).
6. Pelajari perilaku variabel static periTerakhir, dengan mencetak nilainya tiap kali sebuah instance Sprite dimunculkan. Amati bahwa isi dari variabel tersebut milik semua peri yang ada akan selalu berubah tiap kali peri baru ditambahkan.

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

```
Peri Red, jumlah hati: 4
Peri Yellow, jumlah hati: 2
Peri Blue, jumlah hati: 9
Peri Green, jumlah hati: 6
Buah Apple, harga: 50 G, status: Belum dipanen
Buah Orange, harga: 40 G, status: Belum dipanen
Buah Banana, harga: 100 G, status: Belum dipanen
Buah Grape, harga: 80 G, status: Belum dipanen
Buah Papaya, harga: 70 G, status: Belum dipanen
Red telah memanen buah Apple dengan harga 50 G
Red telah memanen buah Orange dengan harga 40 G
Yellow telah memanen buah Banana dengan harga 100 G
Blue telah memanen buah Grape dengan harga 80 G
Blue telah memanen buah Papaya dengan harga 70 G
Green tidak jadi memanen buah Orange karena sudah tidak ada
Buah Apple, harga: 50 G, status: Sudah dipanen
Buah Orange, harga: 40 G, status: Sudah dipanen
Buah Banana, harga: 100 G, status: Sudah dipanen
Buah Grape, harga: 80 G, status: Sudah dipanen
Buah Papaya, harga: 70 G, status: Sudah dipanen
Atas jasa Red dalam memanen sejumlah 90 G, Red telah menerima 10 buah
berry dan 1 buah truffle
Atas jasa Yellow dalam memanen sejumlah 100 G, Yellow telah menerima 10
buah berry dan 3 buah truffle
Atas jasa Blue dalam memanen sejumlah 150 G, Blue telah menerima 60 buah
berry dan 2 buah truffle
Karena Green tidak memanen satu pun buah, Green tidak diberikan bayaran
apapun
```

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)
  - Sprite.java
  - Fruit.java
  - Kelas tambahan seperlunya