Responsi 13

Analisis dan Desain Berorientasi Objek

Interface Segregation Principle

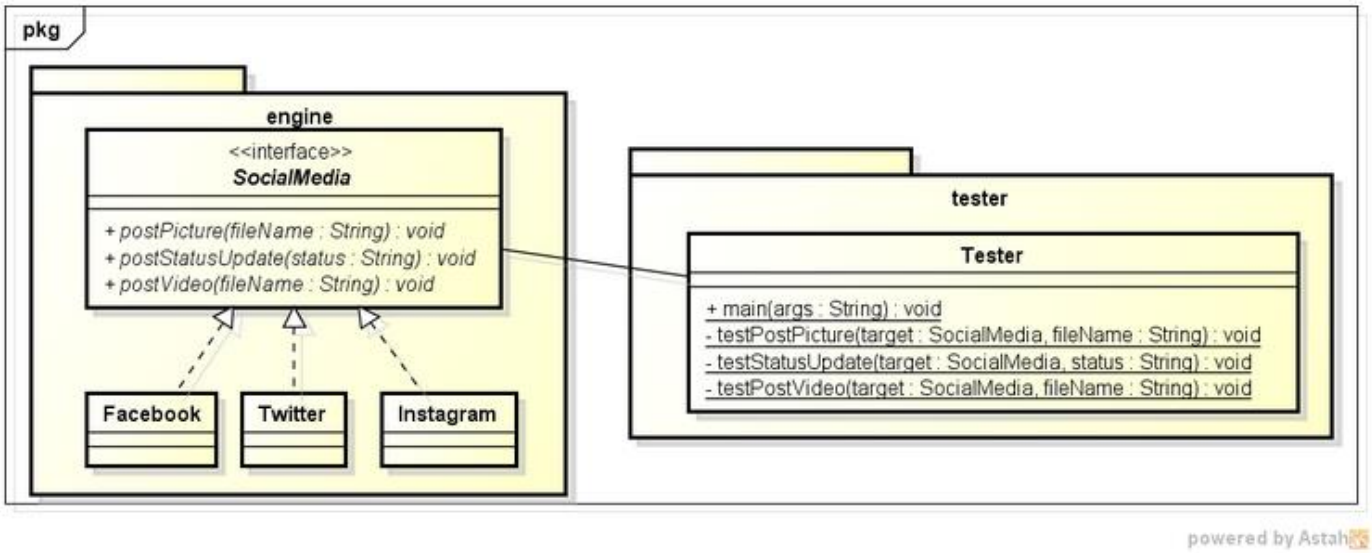
Pada praktikum kali ini, kita akan mencoba prinsip keempat dari SOLID Principles, yaitu Interface Segregation Principle (ISP). Kata kunci dari ISP adalah “no client should be forced to depend on methods it does not use”.

Sebagai seorang programmer, tentunya kita akan memikirkan rancangan kelas-kelas yang akan kita gunakan. Kita pun telah mengenal abstract class dan interface yang dapat berfungsi sebagai pemegang kontrak dalam rancangan program kita. Untuk menentukan abstract class dan interface yang akan digunakan, seorang programmer harus memikirkan abstraksi dari modul-modul yang akan dikerjakan. Tantangan berikutnya, seorang programmer harus memikirkan kemungkinan perubahan yang terjadi pada program tersebut.

Permasalahan yang kadang terjadi adalah programmer belum dapat mendeteksi kemungkinan tambahan-tambahan dalam program, yang hanya akan mengimplementasikan submodul dari program yang sudah ada. Jika ada penambahan pada program yang mengimplementasikan sebuah interface namun ternyata tidak memerlukan seluruh method pada interface tersebut, program tersebut ‘terpaksa’ tetap harus mengimplementasikan seluruh merthod, walau dengan membuat method-method palsu atau kosong. Interface yang seperti itu dapat disebut dengan fat interface atau polluted interface. ISP adalah prinsip yang menekankan penggunaan interface bukan sebagai sebuah interface besar, melainkan dibagi menjadi interface-interface kecil agar modul-modul yang ada dapat memilih interface mana saja yang benar-benar diperlukannya.

R1301xxyyy - Social Media Poster

Diberikan sebuah program (R1301xxyyy.zip) untuk menguji tiga aplikasi media sosial, yaitu *Facebook*, *Twitter*, dan *Instagram*. Masing-masing social media memiliki kemampuan untuk paling tidak satu dari ketiga fitur berikut: *post picture*, *status update*, dan *post video*. Untuk lengkapnya dapat dilihat pada diagram kelas berikut:



Berikut adalah kemampuan spesifik dari masing-masing media social:

a. Facebook
   - postPicture
   - postStatus
   - postVideo

b. Twitter
   - postStatusUpdate

c. Instagram
   - postPicture

Media sosial yang tidak mendukung fitur tertentu (misal: *Twitter* tidak dapat melakukan *post video*) akan melemparkan UnsupportedOperationException.

Saat ini Instagram ingin mengimplementasikan method untuk melakukan editPicture. Tambahkanlah method tersebut pada interface Social Media. Isi dari method tersebut adalah menampilkan pesan "Edit picture with Instagram". Perhatikanlah, kelas apa sajakah yang perlu Anda ubah? Bukankah saat ini Anda menambahkan method-method yang hanya berisi UnsupportedOperationException pada kelas Facebook dan Twitter?

Hal ini dikarenakan kita telah mengimplementasikan *fat interface* atau interface yang terlalu gemuk, sehingga seluruh kelas yang mengimplementasikan interface tersebut harus mengimplementasikan method-method yang ada walau tidak memerlukannya. Struktur *fat interface* tersebut bukanlah struktur yang baik.

Kerugian berikutnya adalah pada saat kompilasi, Anda tidak tahu, method-method mana saja yang sebenarnya tidak di-support oleh media social tertentu karena seluruh method dipaksakan diimplementasikan walau method tersebut sebenarnya tidak berguna.

Perbaikilah struktur kelasnya memanfaatkan *Interface Segregation Principle*, sehingga sudah terlihat pada saat kompilasi, fitur apa saja yang tidak didukung oleh sebuah media sosial. Dengan kata lain:

- Method testPostPicture() hanya dapat menerima media sosial yang mampu melakukan *post picture*.
- Method testStatusUpdate() hanya dapat menerima media sosial yang mampu melakukan *status update*.
- Method testPostVideo() hanya dapat menerima media sosial yang mampu melakukan *post video*.

Jika struktur program Anda sudah benar dan Anda membuka baris-baris yang di-comment pada kelas Tester, maka baris-baris tersebut akan menyebabkan error.

(Soal diadaptasi dari soal UAS ADBO 2013/2014)

R1302xxyyy – Computer

Dalam sistem kerja komputer, kita mengnal istilah Random Access Memory (RAM) yang adalah sebuah perangkat keras komputer yang berfungsi menyimpan berbagai data dan instruksi program. Selain itu ada juga Arithmetic and Logic Unit (ALU) yang merupakan salah satu bagian dari mikroprosesor yang berfungsi untuk melakukan operasi hitungan. Selain itu, tentunya komputer juga memiliki prosesor yang bertugas untuk mengorganisasikan proses-proses yang dijalankan oleh komputer.

- Seperti yang telah diselaskan pada mata kuliah arsitektur komputer, bahwa komputer kita terdiri dari memory yang dapat menampung data. Sebutlah data yang ditampung tersebut sebagai `MemoryEntry`. MemoryEntry akan kita implementasikan sebagai sebuah class yang memiliki atribut `address` dan `value`. Untuk mempermudah masalah, memory yang saat ini digunakan dilambangkan oleh sebuah karakter dengan huruf capital, dari A sampai Z (otomatis akan hanya ada 26 slot memory). Value yang dapat disimpan hanyalah bilangan bulat.
- Memory tersebut ini ditampung oleh RAM. Class `RAM` terdiri dari sebuah variabel yang menampung kumpulan `MemoryEntry` (array of `MemoryEntry`). Class ini mengimplementasikan beberapa method, yaitu:
  1. `void save(char address, int newValue)`
     Method ini berfungsi untuk memasukan `newValue` pada slot memory ke-`address`.
     Jika ternyata memory pada slot ke-`address` sudah memiliki value (tidak kosong atau null), maka value tersebut akan digantikan dengan nilai yang baru (`newValue`).
  2. int load(char address)
     Method ini berfungsi untuk mengembalikan `value` dari memory ke-`address`.
  3. `MemoryEntry[] showFinalVariables()`
     Method ini berfungsi untuk mengembalikan seluruh isi dari memory yang pernah diakses pada simulasi yang dilakukan.

  Note: source code untuk kelas RAM tidak diberikan, silakan Anda implementasikan.

- Pada program kali ini, ALU melaksanakan tugasnya untuk melakukan operasi matematika. Method yang telah diimplementasikan adalah `int add(int var1, int var2)`.

- Seluruh proses tersebut akan dijalankan oleh kelas Computer. Kelas Computer mengimplementasikan method `void processProgram(String[] commandList)`. Method ini menerikma kumpulan command atau perintah yang akan dioperasikan dari kelas `Tester`. Berikut adalah beberapa ketentuan mengenai jenis-jenis command yang ada.
  a. Command untuk melakukan assignment memory dengan sebuah value. Format untuk command ini adalah:

     [address]=[value]

     Penulisan tanpa spasi dan value hanya bernilai antara 0 sampaii 9.
     Contoh: A=5
  b. Command untuk melakukan assignment memory dengan sebuah variabel lain. Format untuk command ini adalah:

     [address1]=[address2]

     Arti dari perintah ini adalah mengisi alamat address1 dengan value yang disimpan pada address2.
     Contoh: C=A
     Arti dari perintah tersebut adalah mwngisi memory pada address C dengan nilai yang tersimpan pada memory dengan address A. Jika dikaitkan dengan keterangan command a, maka nilai memory C saat ini adalah 5.
  c. Command untuk melakukan penjumlahan dua buah bilangan. Format dari command ini adalah:

     [address1]=[address2]+[address3]

     Arti dari perintah tersebut adalah mengisi isi memory ke-address1 dengan hasil penjumlahan isi dari memory ke-address2 dengan isi dari memory ke-address3.
     Contoh: E=A+C
     Jika dikaitkan dengan keterangan command a dan b, maka nilai dari memory ke-E saat ini adalah 10.

Seluruh command dituliskan tanpa spasi.

Untuk menjaga kontrak dari seluruh rangkaian operasi komputer tersebut, seluruh kelas mengimplementasikan interface `CompProcessor`.

Berikut adalah contoh output dari Tester yang diberikan:

```
run:
A = 5
E = 5
C = 10
D = 8
M = 10
Z = 18
BUILD SUCCESSFUL (total time: 0 seconds)
```

Saat ini, tambahkanlah method subtract yang berfungsi untuk melakukan operasi pengurangan dengan format perintah yang sama dengan method add. Tambahkanlah method subtract tersebut pada interface CompProcessor, dan lalukan penyesuaian-penyesuaian yang diperlukan. Tambahkanlah kasus uji pada kelas Tester untuk menguji proses pengurangan tersebut.

…… [kerjakan terlebih dahulu tugas di atas (pengurangan) sebelum melanjutkan mengerjakan soal ini] ……..

Apakah Anda merasa begitu banyak hal yang perlu Anda ubah? Bahkan di kelas yang tidak bersangkutan sekali pun, Anda harus melalukan penyesuaian.

Selesaikanlah soal ini dengan mengubah struktur dari program yang ada dengan memanfaatkan prinsip Interface Segregation Principle. Pada akhirnya, program Anda harus dapat mengimplementasikan penjumlahan, pengurangan, perkalian, dan pembagian. Seluruh operasi dilakukan terhadap bilangan bulat saja, dan hanya terdiri dari maksimal dua operand.

NOTE: Anda tidak diperkenankan mengubah class MemoryEntry.