Dasar-Dasar Pemrograman 2

Tugas Pemrograman 2

GlasDOS

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

GlasDOS

Sumber

Berkat kesuksesan kamu membantu Dek Depe dalam membuat program HammingCode, Salman kini dapat berkomunikasi dengan Dina secara intensif. Namun, hal ini berimbas buruk kepada Dek Depe dan kamu karena Salman menjadi malas untuk mengisi penilaian Lab dan Tugas Pemrograman DDP 2. Akibatnya, Dek Depe dan kamu menjadi tidak bisa melihat hasil penilaian pekerjaan kamu secepatnya.

Ternyata, masalah ini tidak hanya terjadi pada mata kuliah DDP 2. Berita tentang suksesnya program HammingCode kamu telah tersebar ke seluruh penjuru Fasilkom, tak terkecuali koordinator asdos mata kuliah yang lain. Para koordinator tersebut juga menjadi malas karena sibuk berkomunikasi dengan teman-temannya menggunakan program HammingCode kamu. Melihat kondisi tersebut, Dek Depe tidak tinggal diam. Ia punya ide cemerlang yang dapat menyelesaikan permasalahan ini.

Dek Depe membuat rancangan sebuah program bernama Global Asdos Data Organizing System (GlasDOS). Dengan program ini, para asisten dosen (asdos) DDP 2 dapat mengisi penilaian tanpa melalui Salman. Jadi, kamu dapat melihat hasil penilaian Lab dan Tugas Pemrograman kamu secepatnya setelah asdosmu menginputnya ke dalam sistem GlasDOS.

Supaya bisa digunakan untuk asdos mata kuliah lain, komponen penilaian yang dapat dimasukkan ke dalam sistem pun dibuat modular. Dengan begitu, setiap asdos bisa membuat skema penilaian sesuai dengan BRP mata kuliahnya masing-masing. Untungnya, hal ini bisa diimplementasikan dengan mudah apabila kita memanfaatkan Object-Oriented Programming.

Berhubung Dek Depe sudah lelah merancang GlasDOS, ia tidak sanggup lagi untuk lanjut mengimplementasikan sistem tersebut. Dek Depe tahu bahwa kamu sudah belajar mengenai Object-Oriented Programming. Akhirnya, Dek Depe meminta bantuan kepada kamu, seorang *programmer* hebat yang akan menyelamatkan nilai mahasiswa Fasilkom!

Spesifikasi GlasDOS

Berikut adalah dokumen spesifikasi GlasDOS yang telah dirancang oleh Dek Depe. Dek Depe sudah mengerjakan sebagian isi *class* GlasDOS untuk mengolah input, kamu hanya perlu melengkapinya dan mengimplementasikan *class-class* berikut.

Perlu diperhatikan bahwa *method-method* sederhana seperti *getter* yang hanya mengembalikan atribut (tanpa melakukan hal lain) **tidak ditulis** di dokumen ini, tetapi kamu **mungkin** tetap harus mengimplementasikannya **(seperlunya saja)**. Untuk lebih lengkapnya mengenai *method* apa saja yang harus diimplementasikan, silakan lihat bagian-bagian yang ditandai `TODO` pada *starter code*.

Untuk memudahkan pengerjaan tugas ini, Dek Depe juga meminta kamu untuk menggambarkan Class Diagram dalam Unified Modeling Language (UML). Diagram yang kamu buat **minimal** memuat seluruh detail *class* yang **ada pada dokumen ini**. Apabila kamu membuat *method*/*attribute* sendiri selain yang ada pada dokumen ini, harap tambahkan juga ke Class Diagram buatanmu **hanya jika kamu rasa itu dapat memudahkan orang lain memahami implementasi yang kamu buat**. Diagram yang terlalu rumit (atau berlebihan) justru bukanlah diagram yang informatif :)

Untuk referensi mengenai Class Diagram dalam UML, silakan buka pranala ini. Kamu dapat membuat diagram menggunakan draw.io atau alat lainnya (bebas).

GlasDOS

Class untuk program utama. Kamu hanya perlu mengimplementasikan beberapa *private method* untuk membantu jalannya program.

Detail *class* **GlasDOS**:

- `AsistenDosen[] asistenDosen`
- `KomponenPenilaian[] templatSkemaPenilaian`
+ `void main(String[] args)`

AsistenDosen

Seorang **AsistenDosen** diidentifikasi dengan **kode** asdos dan **nama**nya. Ia memiliki daftar berisi sejumlah **mahasiswa** yang menjadi tanggung jawabnya.

Detail *class* **AsistenDosen**:

- `List<Mahasiswa> mahasiswa`
- `String kode`
- `String nama`
+ `Mahasiswa getMahasiswa(String npm)`:

  Mengembalikan objek **Mahasiswa** dari daftar **mahasiswa** yang menjadi tanggung jawab berdasarkan **NPM** yang diberikan.
+ `void addMahasiswa(Mahasiswa` **mahasiswa**`)`:

  Menambahkan **mahasiswa** ke dalam daftar `mahasiswa`. Ketika menambahkan mahasiswa, pastikan daftar mahasiswa tetap **terurut menaik berdasarkan NPM**.
+ `String rekap()`:

  Merekap seluruh mahasiswa yang menjadi tanggung jawabnya. Manfaatkan *method* `toString()` dan `rekap()` pada Mahasiswa.
+ `String toString()`:

  Mengembalikan kode dan nama dalam format berikut.

  ```
  PW - Kak Pewe
  ```

Mahasiswa implements Comparable<Mahasiswa>

Seorang mahasiswa peserta kuliah diidentifikasi dengan **NPM** dan **nama**nya. Ia memiliki serangkaian **KomponenPenilaian** yang berisi nilai-nilai pekerjaannya. Banyaknya **KomponenPenilaian** dalam rangkaian tersebut sudah ditentukan dari awal berdasarkan skema yang dibuat ketika program baru dijalankan.

Detail *class* **Mahasiswa**:

- KomponenPenilaian[] komponenPenilaian
- String npm
- String nama
+ KomponenPenilaian getKomponenPenilaian(String namaKomponen):

  Mengembalikan **KomponenPenilaian** yang ber**nama namaKomponen** milik seorang mahasiswa.

+ int compareTo(Mahasiswa other):

  Dua mahasiswa dibandingkan berdasarkan NPM-nya. Seorang mahasiswa dianggap "sebelum" mahasiswa lain jika NPM-nya lebih kecil daripada mahasiswa lain dan sebaliknya. *Hint*: String juga punya *method* compareTo(String other).

+ String toString():

  Mengembalikan NPM dan nama seorang mahasiswa dengan format berikut.

```
1234567890 - Dek Depe
```

+ String rekap():

```
Rerata Lab: 90.00
Rerata TP: 95.00
Nilai akhir: 93.00
Huruf: A
LULUS
```

`+ String getDetail():`

Mengembalikan detail setiap komponen penilaian seorang mahasiswa dengan format berikut.

```
~~~ Lab (40%) ~~~
Lab 1: 72.00 (T)
Lab 2: 100.00
Lab 3: 64.00 (T)
Rerata: 78.67
Kontribusi nilai akhir: 31.47

~~~ TP (60%) ~~~
TP: 72.00 (T)
Kontribusi nilai akhir: 43.20

Nilai akhir: 74.67
Huruf: B
LULUS
```

KomponenPenilaian

Beberapa **KomponenPenilaian** akan digunakan dalam menghitung nilai akhir. Sebuah **KomponenPenilaian** terdiri atas beberapa **ButirPenilaian** yang banyaknya sudah ditentukan sejak awal berdasarkan skema penilaian. **KomponenPenilaian** memiliki **nama** sebagai label komponen dan **bobot** yang menandakan persentase kontribusi komponen terhadap nilai akhir. Asumsikan nama komponen tidak memuat spasi memuat spasi di dalamnya.

`Detail` *class* **KomponenPenilaian**:

- `String nama`
- `ButirPenilaian[] butirPenilaian`

  Ketika objek `KomponenPenilaian` dibuat, `butirPenilaian` hanya berupa array `ButirPenilaian[]` dengan panjang tertentu yang belum diisi dengan objek `ButirPenilaian` sama sekali. Artinya, isinya `[null, null, …]`, **bukan** objek-objek `ButirPenilaian` dengan nilai `0`.
- `int bobot`

`+ void masukkanButirPenilaian(int` **idx**`, ButirPenilaian` **butir**`):`

Mengisi **butirPenilaian** pada index ke-**idx** dengan **butir**.

`+ double getRerata():`

Mengembalikan rata-rata `ButirPenilaian` pada `KomponenPenilaian`. Rata-rata dihitung dengan menjumlahkan setiap nilai butir dan membagi jumlahan tersebut dengan **banyaknya butir yang sudah diinput oleh asdos (bukan banyaknya butir untuk komponen ini yang telah ditentukan pada skema penilaian awal)**.

Harap perhatikan bahwa butir dengan nilai `0` berbeda dengan "belum diinput" (hint: belum diinput berarti `null`). Apabila belum ada satu pun yang diinput oleh asdos, maka kembalikan **`0`**.

`+ double getNilai():`

Mengembalikan rata-rata butir penilaian yang sudah dikalikan dengan bobot penilaian.

`+ String toString():`

Mengembalikan rata-rata butir penilaian pada komponen ini.

```
Rerata Lab: 90.00
```

`+ String getDetail():`

Mengembalikan detail yang berisi tiap **`ButirPenilaian`** (yang sudah diinput) dalam komponen ini, rata-rata butir, dan kontribusi komponen terhadap nilai akhir. Setiap butir penilaian diberi nomor urut (dimulai dari 1). Manfaatkan *method* **`toString`** dari **`ButirPenilaian`**.

```
~~~ Lab (40%) ~~~
Lab 1: 80.00 (T)
Lab 2: 100.00
Rerata: 90.00
Kontribusi nilai akhir: 36.00
```

Apabila hanya terdapat 1 `ButirPenilaian` (berdasarkan skema, bukan berdasarkan yang sudah diinput asdos) pada suatu `KomponenPenilaian`, maka jangan berikan nomor urut maupun rerata.

```
~~~ Partisipasi (2%) ~~~
Partisipasi: 90.00
Kontribusi nilai akhir: 1.80
```

ButirPenilaian

ButirPenilaian merupakan komponen terkecil dalam penilaian. Suatu **ButirPenilaian** memiliki **nilai** dan indikator apakah pekerjaan terkait butir tersebut dikumpulkan **terlambat** atau tidak. Apabila **terlambat**, nilai yang ada pada suatu **ButirPenilaian** akan dikurangi sebesar **20 persen dari nilai yang sebenarnya**.

Detail *class* **ButirPenilaian**:

- `double nilai` (hanya boleh >= 0, jika < 0 set menjadi 0)
- `boolean terlambat`
+ `double getNilai()`:

  Mengembalikan nilai butir yang sudah disesuaikan dengan keterlambatan (jika **terlambat**).
+ `String toString()`:

  Mengembalikan nilai butir ini. **Jika `terlambat`, beri penanda** `(T)`.

```
80.00 (T)
```

Alur Program

Contoh alur program dapat diakses pada pranala berikut.

Catatan Khusus

Berikut merupakan beberapa catatan khusus mengenai pengerjaan program.

- Kamu **tidak perlu** menangani kasus yang tidak tertulis pada dokumen ini **dan** tidak dicontohkan pada kasus uji yang disediakan pada direktori testcases. Beberapa contoh di antaranya:
    - Input tidak sesuai format.
    - Lebih dari satu NPM yang sama untuk `AsistenDosen` yang sama maupun berbeda.
    - Lebih dari satu kode yang sama untuk `AsistenDosen` yang berbeda.
    - Lebih dari satu `KomponenPenilaian` dengan nama yang sama pada `Mahasiswa` yang sama.
    - Total `bobot` komponen penilaian >100 persen (coba cek BRP :D).
- Kasus tepi yang perlu kamu tangani meliputi (tetapi tidak terbatas pada):
    - Asdos menginput nilai butir penilaian secara tidak berurutan dan/atau tidak lengkap.
    - Asdos dapat mengisi ulang butir penilaian yang sama (akan menimpa penilaian yang lama).
    - Asdos tidak mengisi satu pun `ButirPenilaian` yang terdapat pada `KomponenPenilaian` seorang mahasiswa.
    - Data `Mahasiswa` yang dimasukkan tidak terurut NPM, data yang dicetak tetap harus terurut menaik berdasarkan NPM.
    - Kasus-kasus lain yang dicontohkan pada kasus uji yang disediakan.
- Kamu boleh menggunakan *library* untuk melakukan pengurutan.
- Yang dimaksud dengan skema penilaian adalah `KomponenPenilaian[]` dengan panjang tertentu yang berisi `ButirPenilaian[]` yang masih kosong.
- Di luar hal-hal tersebut, silakan **buat asumsi** (selama tidak bertentangan dengan soal) dan jelaskan kepada `AsistenDosen` ketika kamu mendemonstrasikan program. Apabila ada hal yang kamu rasa benar-benar perlu diklarifikasi, silakan tanyakan melalui Forum Tugas Pemrograman di SCeLE, GitLab Issues pada repositori pusat, atau `AsistenDosen` kamu.

Bonus (5%)

Implementasikan menu ke-4 untuk "`Ubah kode asdos mahasiswa`". Menu tersebut menerima input dalam format berikut.

`[KodeAsdosAsal] [NPM] [KodeAsdosBaru]`

Contoh:

`SMA 1234567890 SMP`

Artinya, `Mahasiswa` dengan `NPM` **`1234567890`** yang semula di bawah tanggung jawab asdos dengan kode **`SMA`** dipindahkan ke dalam daftar mahasiswa milik asdos dengan kode **`SMP`**. Mahasiswa tersebut tidak ada lagi di dalam daftar mahasiswa milik asdos dengan kode **`SMA`**. Namun, seluruh **`KomponenPenilaian`**-nya tetap seperti semula (tidak ada yang berubah).

Kamu perlu memeriksa bahwa `Mahasiswa` tersebut memang sebelumnya ada pada daftar milik `AsdosAsal`. Apabila `Mahasiswa` tersebut tidak ditemukan pada daftar milik `AsdosAsal`, keluarkan output **`"Mahasiswa tidak ditemukan!"`**.

Kamu juga perlu memeriksa bahwa `AsdosAsal` maupun `AsdosBaru` telah terdaftar dalam sistem. Jika `AsdosAsal` tidak ditemukan, tuliskan **`"Asdos dengan kode [KodeAsdosAsal] tidak ditemukan!"`**. Jika AsdosBaru tidak ditemukan, tuliskan **`"Asdos dengan kode [KodeAsdosBaru] tidak ditemukan!"`**. Apabila keduanya tidak ditemukan, tuliskan **`"Asdos dengan kode [KodeAsdosAsal] dan [KodeAsdosBaru] tidak ditemukan!"`**.

Akhiran

Untuk mengetahui kebenaran program yang kamu buat, ada dua cara yang dapat dilakukan. Pertama, coba tiap kasus yang ingin kamu uji menggunakan program main yang telah disediakan. Asdos sudah menyediakan beberapa kasus uji yang terdapat pada direktori `testcases`. Berkas dengan awalan `in` berarti untuk input, dan berkas dengan awalan `out` berarti untuk output.

Cara yang lain, kamu dapat menjalankan

```
# di Windows
> gradlew.bat :assignment2:test
# di Linux/Mac
$ ./gradlew :assignment2:test
```

dan melihat apakah ada test yang gagal. Perlu diperhatikan bahwa *test* yang disediakan untuk dijalankan dengan Gradle hanya menguji nilai kembalian (*return*) dari beberapa *method* yang krusial saja dan bukan *output* program secara keseluruhan. Hal ini dibuat sedemikian rupa untuk menghindari *test* yang gagal karena perbedaan *output* yang minor. Kamu disarankan untuk juga menguji input dan *output* program berdasarkan *test case* yang ada pada direktori `testcases`.

Kumpulkan Class Diagram dalam format PDF (ukuran kertas bebas) dengan format penamaan `TP2_CD_[Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap].pdf`. Contoh:

`TP2_CD_A_SMP_1234567890_DekDepe.pdf`

Penilaian:

10% Class Diagram dalam Unified Modeling Language (UML)
50% kebenaran program
30% demonstrasi program
10% kerapian dan dokumentasi

+5% implementasi bonus