TUGAS PEMROGRAMAN 2

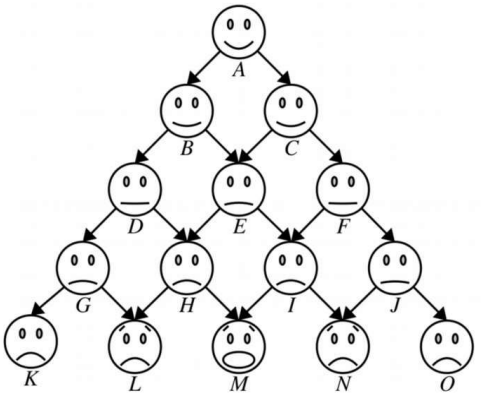
Dasar dasar pemrograman 2 - Genap 2018/2019

Pembuka

Pada suatu hari di Fasilkom, diadakanlah suatu acara yang bernama Computer Get Together (CGT). Di acara CGT tersebut, banyak event-event yang ditujukan untuk memperdalam kekompakan elemen Fasilkom. Salah satu event yang diadakan adalah tantangan untuk membuat piramid yang dibentuk dari elemen-elemen Fasilkom.

Sebagai mahasiswa tingkat pertama, anda tentu sangat antusias untuk mengikuti kegiatan ini. Tidak lama setelah acara diumumkan, seluruh kelas DDP 2 tahun 2019 sepakat untuk berpartisipasi. Pertanyaan berikutnya adalah : bagaimana menyusun piramid manusia tersebut? Untuk penyusunan yang efektif, anda meminta kakak kelas anda yang sudah mengambil kuliah SDA dan DAA untuk mencarinya. Sedangkan tugas anda adalah mencari tahu, **seberapa berat beban yang harus dipikul oleh setiap orang di setiap bagian piramid tersebut**.

Deskripsi Masalah



Piramid manusia adalah suatu bentuk penyusunan/formasi manusia secara vertikal dan dibuat agar membentuk piramid. Masing-masing baris memiliki jumlah orang yang berbeda dengan baris paling atas berisi 1 orang, dibawahnya 2 orang, dan seterusnya. Gambar di samping ini adalah ilustrasi piramid manusia dengan **5 baris**.

Setiap orang memiliki berat badan yang harus dipikul oleh orang dibawahnya. Asumsikan setiap orang (kecuali baris paling bawah) dipikul oleh tepat 2 orang, sehingga beban yang harus dipikul oleh 2 orang tersebut sama rata, yaitu berat badan orang yang dipikul dibagi 2. Sebagai contoh, misalkan berat badan A adalah 60kg. Maka, B akan menanggung berat badan A sebesar 30kg. Hal yang sama juga terjadi pada C yang juga menanggung beban berat badan A sebesar 30kg.

Untuk kasus baris berikutnya (orang D, E, dan F), tentu mereka harus menanggung tidak hanya berat badan B dan C, tetapi juga beban yang dipikul oleh B dan C. Dalam kasus seperti ini, jumlahkan saja berat badan orang yang dipikul beserta beban yang dipikul oleh orang tersebut. Asumsikan berat badan B adalah 50kg dan berat badan C 70kg. Maka beban yang dipikul oleh D adalah ½ * (50kg + 30kg), yaitu **40kg**. Beban yang dipikul oleh E adalah ½ * (50kg + 30kg) + ½ * (70kg + 30kg), yaitu **90kg**. Apakah kamu bisa melihat polanya?

Secara umum, masalah yang kamu hadapi dapat dirumuskan seperti ini :

- Setiap orang memiliki berat badan dan beban yang harus dipikul
- Setiap orang memikul beban sebesar setengah dari berat badan orang yang dipikul, ditambah dengan setengah dari beban orang yang dipikul.

Detil Teknis

Anda diberikan template program yang **sangat dianjurkan untuk diubah** sesuai dengan kebutuhan anda. Anda tidak harus menggunakan template ini selama anda yakin dengan program anda. Program terdiri dari 2 kelas yang harus ditambahkan/diimplementasi yaitu kelas `Pyramid` dan `Person`. Kelas `Pyramid` juga mengandung method main sebagai kelas utama. Pada kelas `Pyramid`, representasi piramid manusia disimpan dalam bentuk *ragged array* dengan nama variable `pyramidData`. Sebagai contoh, `pyramidData`[0][0] akan menyimpan informasi `Person` A, `pyramidData`[1][1] akan menyimpan informasi `Person` C, dan seterusnya.

Untuk kelas `Person`, anda perlu mengimplementasikan `getTotalWeight()` yang akan mengembalikan total berat badan (`weight`) ditambah dengan beban (`load`) suatu `Person`. Anda juga perlu untuk mengimplementasi method getter (*aksesor*) dan setter (*mutator*) yang sesuai untuk keperluan anda. Anda juga mungkin perlu untuk melakukan implementasi method `toString()`.

Dalam mencari beban yang harus dipikul oleh suatu Person, anda harus mengimplementasi fungsi `double computeLoad(int row, int col)` yang akan mengembalikan beban (load) pada suatu baris dan kolom. Baris dan kolom mengacu pada index yang ada di variable `pyramidData`. Pada contoh di atas, `computeLoad(0,0)` akan mengembalikan 0, `computeLoad(1,1)` mengembalikan 30, dan `computeLoad(2,1)` mengembalikan 90. Pengimplementasian method ini **harus dilakukan secara rekursif dan tidak menggunakan ArrayList**.

Pada template y ang diberikan, program yang anda buat harus bisa menerima 2 macam argumen saat dijalankan. Jika argumen adalah bilangan bulat, maka gunakan fungsi untuk membuat piramid secara random menggunakan method `fillData(int level)`. Selain itu, maka argumen akan dianggap berkas yang dibaca sebagai representasi piramid yang ada di direktori yang sama dengan program anda. Format berkas sebagai berikut :

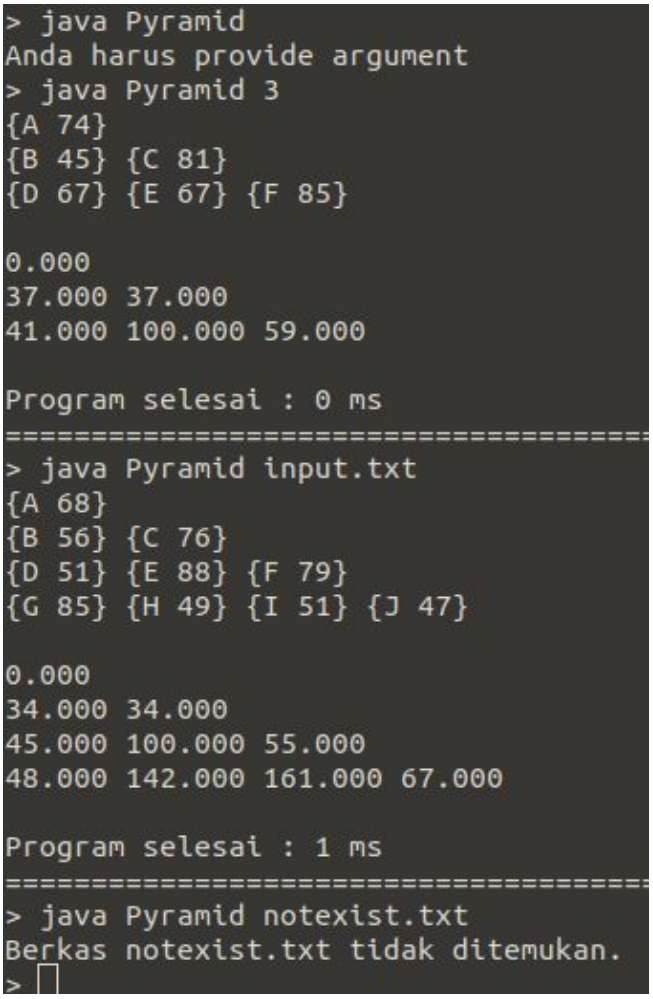
```
<jumlah level>
Nama1 weight1;
Nama2 weight2;Nama3 weight3;

dst..
```

Contoh input:

```
4
A 68;
B 56;C 76;
D 51;E 88;F 79;
G 85;H 49;I 51;J 47;
```

Selain itu, anda juga harus mengecek performa program anda. Fungsi/contoh cara pengecekan performa juga sudah tersedia di template program.

Contoh eksekusi program



Jika tidak menerima input apapun, program tidak berjalan

Untuk **level = 3,** 3 baris pertama adalah informasi nama dan berat badan `Person` yang direpresentasikan dengan array. 3 Baris berikutnya berisi beban yang dipikul oleh suatu Person di indeks tersebut. Contohnya, pada kasus ini E akan memikul beban seberat 100. Baris terakhir berisi lamanya program ini dijalankan.

Untuk input berupa file, maka isi piramid akan menyesuaikan dengan file tersebut. Isi dari file tersebut diasumsikan sesuai format. File input.txt di samping berisi contoh yang diberikan sebelumnya.

Untuk setiap eksekusi program, harus ada waktu yang menunjukkan berapa lama program tersebut berjalan.

Jika file/berkas tidak ditemukan, program anda juga tidak dapat berjalan

Teknis pengumpulan

Anda harus mengumpulkan file .java yang berisi kode sumber Anda. Selain itu, anda juga diminta membuat Executable Jar File .jar yang dihasilkan dari program Anda. Simpan kode sumber .java dan Executable Jar File .jar pada sebuah file .zip.

Kumpulkan sebuah file .zip yang berisi:

1. Sebuah file .java berisi kode sumber program buatan Anda yang diberikan komentar.
2. Sebuah executable file .jar untuk menjalankan program yang Anda buat

dengan format penamaan **[Kode Asdos]_[Nama]_[Kelas]_[NPM]_TP2.zip** pada slot di **SceLE** paling lambat **Jumat, 5 April 2019 pukul 18.00 WIB**.

Komponen penilaian

50% : correctness (kebenaran) program
25% : pembuatan kelas dan method sesuai OO
10% : metode penyimpanan data yang optimal
10% : dokumentasi
5% : implementasi fitur atau method tambahan

Komponen bonus

Untuk setiap pemanggilan fungsi rekursif `computeLoad` , besar kemungkinan fungsi tersebut akan memanggil 2 fungsi `computeLoad` lain. Hal ini akan berdampak pada lamanya waktu program anda jika *level* yang diberikan lebih dari 20. Anda dapat bandingkan dan catat waktu yang dibutuhkan untuk *level* yang berbeda, misalnya 5, 15, 35, dan 50.

Untuk mempercepat proses perhitungan seluruh beban, anda dapat menggunakan teknik yang dinamakan ***memoization***. Teknik ini menyimpan nilai sementara dari pemanggilan rekursif anda sehingga tidak perlu dilakukan perhitungan hingga titik paling awal.

Untuk memeroleh **poin bonus** (10%), lakukan hal berikut :

- Implementasi teknik ***memoization*** pada program anda di fungsi yang berbeda, namun memiliki header yang sama. Dalam hal ini, implementasikan fungsi `double computeLoadMemo(int row, int col)`.
- Pastikan correctness dari fungsi yang baru anda buat dengan fungsi sebelumnya benar.
- Anda juga harus memberikan opsi bagi pengguna untuk menggunakan fitur ini, tentu dengan mengubah argumen program java anda. Formatnya dibebaskan.
- Bandingkan waktu yang diperlukan pada *level* yang berbeda (5, 15, 35, dan 50) dengan implementasi fungsi sebelumnya.
- Laporkan penemuan anda dan tulis dalam dokumen teks yang diikutsertakan pada berkas .zip yang anda kumpulkan.

SELAMAT BELAJAR DAN SEMANGAT MENGERJAKAN
