TUGAS 1

Dasar-Dasar Pemrograman 2
CSGE601021
Semester Genap 2016/2017

Batas waktu pengumpulan:
Senin, 24 Maret 2017 pukul 23.55 Waktu Scele

Tujuan dari Tugas ini adalah untuk menguji pemahaman Anda terhadap materi bahan kuliah yang telah diajarkan di kelas. Mahasiswa **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!

TUGAS 1

Panitia penyelenggara CS League meminta bantuan Anda sebagai *programmer* untuk membangun sistem yang mengatur tentang kompetisi di liga fasilkom. Anda diminta untuk membuat sistem yang mengakomodasi kebutuhan panitia tersebut. *Requirement* yang diberikan adalah sebagai berikut:

- Sistem kompetisi yang digunakan ialah sistem setengah kompetisi (*round robin*). Sehingga, setiap tim harus bertanding satu dengan yang lainnya untuk memperebutkan peringkat teratas. Jika jumlah Tim ialah sebanyak *t* akan terdapat C_2^t pertandingan.
- Urutan peringkat dalam klasemen diatur dengan aturan sebagai berikut:
    - Menang: +3 poin
    - Kalah: 0 poin
    - Seri: +1 poin
- Jika pada akhir kompetisi terdapat Tim dengan poin yang sama, maka, pemenang akan ditentukan dengan melihat selisih gol (jumlahGol – jumlahKebobolan) terbanyak.

Sebelum sistem ini diterapkan, Anda diminta untuk membuat *prototype* sistemnya. *Prototype* sistem tersebut dapat membuat representasi data *dummy* untuk Pemain, Tim, hingga Pertandingan (*Game*). Karena masih hanya sebuah *prototype*, maka jumlah tim *t* yang diperbolehkan pun dibatasi yaitu 4 sampai dengan 10 ($4 \leq t \leq 10$) dengan masing-masing Tim terdiri dari tepat 5 pemain.

Setelah berpikir semalaman suntuk, akhirnya Anda memutuskan untuk menggunakan konsep *Object Oriented Programming* (OOP) untuk memodelkan permasalahan tersebut. Anda kemudian mulai membuat *draft* kelas-kelas dan *field* apa yang akan anda gunakan, yaitu sebagai berikut:

1. Kelas **Pemain**
   Kelas ini menyimpan informasi tentang Pemain. Informasi utama yang akan Anda simpan ialah nomorPemain dan nama namaPemain. Selain itu, Anda juga akan menyimpan informasi jumlahGol, jumlahPelanggaran, jumlahKartuKuning, dan jumlahKartuMerah dalam kelas ini.
2. Kelas **Tim**
   Kelas ini menyimpan informasi Tim. Seperti telah disebutkan sebelumnya, setiap tim terdiri dari 5 Pemain. Sejenak, Anda pun teringat tentang konsep array dan ArrayList yang sudah pernah diajarkan di kelas dan Anda memutuskan untuk menyimpan kelima Pemain tersebut dalam sebuah array bertipe Pemain. Di dalam kelas Tim, Anda juga akan menyimpan informasi namaTim, peringkat, jumlahMenang, jumlahKalah, jumlahSeri, jumlahPoin, jumlahGol, dan jumlahKebobolan.
3. Kelas **Liga**
   Kelas ini menyimpan data klasemen, yaitu, array dari Tim yang berurutan (*sorted*) yang urutannya akan bergantung pada posisi dalam klasemen. Setiap sebuah pertandingan selesai, array ini akan di-*update* berdasarkan poin yang diperoleh masing-masing tim atau selisih gol terbanyak jika ada dua tim yang memiliki poin sama.
4. Kelas **Game**
   Kelas ini menyimpan informasi dan *method* yang terkait dengan permainan. Dalam *draft*, Anda baru hanya mencatat dua *field* yang akan disimpan di kelas ini, yaitu *field* yang menyimpan informasi tentang Liga dan field isFinished yang menyimpan informasi apakah Liga telah selesai atau belum. Namun, Anda tahu bahwa kedepannya Anda akan menambahkan beberapa *field* dan *method* dalam kelas ini.

Uraian diatas hanya merupakan *draft* awal yang masih harus Anda lengkapi lagi. Anda boleh menambahkan (tapi jangan mengurangi) *field* atau *method* berdasarkan kebutuhan dan *requirement* sistem. Anda pun diperbolehkan menambahkan kelas Baru jika memang diperlukan. Namun perlu diingat, jika menambahkan kelas baru, Anda harus berpikir tentang tingkat *cohesion* dari kelas tersebut.

Dalam *prototype* yang akan Anda buat, sistem akan terus meminta perintah (*command*) kepada *user*. Sistem baru akan berhenti meminta input jika Game sudah selesai yang ditandai dengan flag isFinished yang bernilai true. Berikut adalah algoritma yang bisa Anda gunakan:

```
while(!game.isFinished()){
        //ask some input
        …
        //parse input
        …
        //execute input (if no errors)
        …
        //notify (if there's error)
        …
}
```

Sistem akan meminta perintah kepada user sebagai berikut:

```
WELCOME…
[LIGAF12]=> _
```

Perintah yang harus Anda buat adalah sebagai berikut:

1. Perintah: **`init x`**
   Perintah ini hanya dapat digunakan pada saat awal sistem dijalankan. Dengan perintah ini, sistem akan menginisiasi Tim yang akan ikut dalam kompetisi dengan data *dummy*. Variabel *x* menyatakan banyaknya tim t_i yang akan dibentuk dengan *range* yang telah disebutkan sebelumnya, yaitu $(4 \leq t \leq 10)$. Dalam menginisiasi tim, ada dua hal yang harus diisi, yaitu, namaTim daftar Pemain. Nama tim dibuat dengan mengambil secara acak (dengan tidak ada nama tim yang sama) dari 18 nama berikut:

   ```
   arrayOfTeamName = ["Gajah", "Rusa", "Belalang", "Kodok", "Kucing",
   "Tupai", "Rajawali", "Siput", "Kumbang", "Semut", "Ular", "Harimau",
   "Kuda", "Serigala", "Panda", "Kadal", "Ayam", "Bebek"]
   *copy array diatas
   ```

   Sebuah tim terdiri dari tepat 5 Pemain. Pemain-pemain tersebut juga diinisiasi dengan data *dummy*. Ada dua hal yang harus diisi saat menginisiasi pemain, yaitu, nomorPemain dan namaPemain. nomorPemain didapatkan secara acak dari nilai dalam rentang 1-99 dengan

tidak ada nomor pemain yang sama dalam satu tim. Sedangkan, namaPemain dapat diambil secara acak, dengan tidak ada nama pemain yang sama dalam satu tim (jika beda tim boleh sama), dari 80 nama berikut:

```
arrayOfPlayerName = [“Arnold”, “Kaidou”, “Chopper”, “Pica”, “Enel”, “Zoro”, “Pedro”, “Beckman”, “Ace”, “Shiryu”, “Sakazuki”, “Marco”, “Garp”, “Dadan”, “Sengoku”, “Sanji”, “Magellan”, “Dragon”, “Sabo”, “Smoker”, “Luffy”, “Franky”, “Borsalino”, “Buggy”, “Crocodile”, “Shanks”, “Yasopp”, “Coby”, “Burgess”, “Usopp”, “Law”, “Kid”, “Bege”, “Yonji”, “Doffy”, “Edward”, “Mihawk”, “Shanks”, “Jinbei”, “Killer”, “Robin”, “Roger”, “Shiki”, “Rayleigh”, “Robb”, “Kuma”, “Moriah”, “Teach”, “Pagaya”, “Conis”, “Hachi”, “Brook”, “Kinemon”, “Vergo”, “Caesar”, “Momo”, “Mohji”, “Cabaji”, “Jozu”, “Vista”, “Doma”, “Augur”, “Drake”, “Ivankov”, “Charlotte”, “Bellamy”, “Demaro”, “Dorry”, “Brogy”, “Kuro”, “Zeff”, “Gin”, “Pearl”, “Alvide”, “Apoo”, “Kuzan”, “Nami”, “Brook”, “Hancock”, “Koala”]
```

***copy array diatas**

2. Perintah: **`nextGame`**
   Perintah ini hanya dapat digunakan setelah perintah **`init`** dilaksanakan. Perintah **`nextGame`** merupakan *core* dari *prototype* yang akan dibuat. **`nextGame`** akan membuat data *dummy* tentang informasi pertandingan, yaitu:
   - Gol
     Perintah **`nextGame`** akan membuat data *dummy* gol secara random. Setiap gol yang dibuat akan disimpan dalam akumulasi jumlahGol pada kelas Pemain (yang mencetak gol). Untuk membatasi fungsi *random* dalam membuat data dummy untuk jumlah gol, maka diberlakukan batasan jumlah gol pada setiap tim, yaitu, $0 \leq jumlahGol \leq 5$
   - Kartu Kuning
     Perintah **`nextGame`** juga membuat data dummy jumlah kartu kuning secara *random*. Batasan jumlah kartu kuning pertim ialah $0 \leq jumlahKartuKuning \leq 3$. Data ini akan diakumulasi dengan data *jumlahKartuKuning* dalam kelas Pemain (yang mendapat kartu kuning).
   - Kartu Merah
     Perintah **`nextGame`** juga membuat data dummy jumlah kartu merah secara *random*. Batasan jumlah kartu merah pertim ialah $0 \leq jumlahKartuMerah \leq 2$. Data ini akan diakumulasi dengan data *jumlahKartuMerah* dalam kelas Pemain (yang mendapat kartu merah).
   - Pelanggaran
     Perintah **`nextGame`** juga membuat data dummy jumlah pelanggaran secara *random*. Batasan jumlah pelanggaran pertim ialah $0 \leq jumlahPelanggaran \leq 5$. Data ini akan diakumulasi dengan data *jumlahPelanggaran* dalam kelas Pemain (yang melakukan pelanggaran).

   Untuk mempermudah proses *testing*, maka dalam perintah **`nextGame`** dibuat mode manual. Mode manual digunakan untuk memasukkan informasi pertandingan secara manual. Untuk mengaktivasi mode manual tersebut, user hanya perlu menambah beberapa argumen tambahan, sebagai berikut:

- Gol (**-g**)
  Untuk mengaktifkan mode manual, argumen **-g** merupakan argumen wajib yang harus ada. Jika tidak ada argumen ini, maka mode manual tidak bisa dijalankan atau dianggap error. Mode manual dengan argumen **-g** akan menambahkan (+1) jumlahGol ke dalam objek Pemain.

  Sintaks:
  ```
  nextGame -g [namaTim] [nomorPemain]
  ```

  Contoh
  ```
  nextGame -g Bebek 1
  ```

  Tentunya, namaTim yang digunakan ialah nama tim yang memang akan bertanding, dan nomorPemain haruslah pemain yang terdaftar dalam tim tersebut. Jika tidak, maka pesan error dimunculkan. Untuk mendapatkan nama tim yang akan bertanding, dapat digunakan perintah **show nextGame** yang akan dijelaskan pada bagian selanjutnya.

  Argumen **-g** dapat digunakan lebih dari satu kali, tentunya, karena dalam pertandingan bisa terdapat lebih dari satu gol. Contoh penggunaan argumen **-g** lebih dari satu kali adalah sebagai berikut:

  ```
  nextGame -g Bebek 3 -g Ular 10 -g Ular 10
  ```

  Hasil dari perintah tersebut ialah skor 2:1 dengan kemenangan untuk tim Ular.

- Kartu Kuning (**-kk**)
  Infomasi selanjutnya yang juga bisa menggunakan mode manual ialah Kartu Kuning. Argumen untuk kartu kuning yaitu **-kk**. Argumen ini tidak bersifat wajib dalam mode manual. Mode manual dengan argument **-kk** akan menambahkan (+1) jumlahKartuKuning ke dalam objek Pemain.

  Sintaks:
  ```
  nextGame -kk [namaTim] [nomorPemain]
  ```

  Contoh
  ```
  nextGame -kk Bebek 1
  ```

- Kartu Merah (**-km**)
  Selanjutnya adalah Kartu Merah. Sama seperti Kartu Kuning, argumen ini tidak bersifat wajib. Mode manual dengan argumen **-km** akan menambahkan (+1) jumlahKartuMerah ke dalam objek Pemain.

  Sintaks:
  ```
  nextGame -km [namaTim] [nomorPemain]
  ```

  Contoh
  ```
  nextGame -km Bebek 1
  ```

Hal yang perlu menjadi catatan ialah Kartu Merah juga bisa didapat jika seorang pemain yang sama mendapat dua Kartu Kuning dalam satu pertandingan.

- Pelanggaran (**-p**)
  Selanjutnya adalah Pelanggaran. Argumen ini juga tidak bersifat wajib. Mode manual dengan argumen **-p** akan menambahkan (+1) jumlehPelanggaran ke dalam objek Pemain.

  Sintaks:
  ```
  nextGame -p [namaTim] [nomorPemain]
  ```

  Contoh
  ```
  nextGame -p Bebek 1
  ```

  Hal yang perlu menjadi catatan ialah Pelanggaran juga akan bertambah secara otomatis jika seorang pemain mendapat Kartu Kuning atau Kartu Merah dalam satu pertandingan.

Jika mode manual diaktifkan, maka fungsi random tidak dijalankan dan tidak diberlakukan batasan untuk jumlah gol, kartu kuning, kartu merah, dan pelanggaran. Mode manual dapat dijalankan secara berbarengan, seperti berikut:

```
nextGame -g Ular 10 -kk Bebek 5 -kk Bebek 5
```

Perintah diatas mencatat statistik yaitu kemenangan 1:0 untuk tim Ular. Tim Bebek melakukan 2 kali pelanggaran, dan 2 kali kartu kuning. Karena yang mendapat 2 kartu kuning bernomor sama, maka no 5 juga mendapat kartu merah.

Selain mode manual diatas, ada satu argument penting yang juga harus dibuat dalam *prototype* ini, yaitu -all. Argumen tersebut berfungsi untuk menyelesaikan seluruh pertandingan dalam kompetisi hingga ditemukan pemenangnya.

Sintaks:
```
nextGame -all
```

Setelah sebuah pertandingan usai, klasemen (urutan Tim) dalam kelas Liga akan di-*update* (array tim di kelas liga akan di-*rearrange*) berdasarkan poin atau selisih gol (jika poinnya sama) sedemikian rupa sehingga Tim dengan poin tertinggi akan menempati array dengan indeks terendah. Selain itu, setelah pertandingan dilaksanakan akan ditampilkan statistika pertandingan sebagai berikut:

```
Statistika Pertandingan Tim Ular VS Tim Bebek

Tim : Ular
Gol : 2
Pelanggaran : 4
Kartu Kuning : 2
Kartu Merah : 1

Tim : Bebek
Gol : 1
Pelanggaran : 2
```

```
Kartu Kuning : 1
Kartu Merah : 0
```

Informasi statistik tersebut hanya disampaikan setiap pertandingan selesai dan tidak perlu disimpan.

3. Perintah: **show**
   Perintah ini berguna untuk menyampaikan informasi terkini terkait pertandingan. Perintah **show** membutuhkan argumen. yaitu:
   - **show klasemen**
     Perintah ini berfungsi untuk menampilkan klasemen sementara terkini. Klasemen ditampilkan berurutan mulai dari puncak klasemen hingga tim dengan *score* terendah. Output dari perintah **show klasemen** adalah sebagai berikut

```
Peringkat | Nama Tim | Jumlah Gol | Jumlah Kebobolan | Menang | Kalah | Seri | Poin
-------------------------------------------------------------------------------------
    1     |   Ular   |     12     |        2         |   12   |   0   |   0  |  36
    2     |   Gajah  |     11     |        2         |   11   |   0   |   0  |  33

    Dst…
```

   - **show pencetakGol**
     Perintah ini berfungsi untuk menampilkan 10 pencetak gol terbanyak. Daftar pencetak gol ditampilkan berurutan mulai tertinggi hingga terendah. Output dari perintah **show pencetakGol** adalah sebagai berikut

```
Peringkat | Nama Pemain | Nama Tim | Jumlah Gol
-----------------------------------------------
    1     |     Luffy   |  Bebek   |     31
    2     |     Zoro    |  Bebek   |     29
Dst…
```

   - **show tim [namaTim]**
     Perintah ini berfungsi untuk menampilkan 5 pemain dalam tim lengkap dengan informasinya. Output dari perintah **show tim Bebek** adalah sebagai berikut

```
Nomor | Nama | Gol | Pelanggaran | Kartu Kuning | Kartu Merah
-------------------------------------------------------------
  7   | Enel |  5  |      5      |      2       |      0
  13  | Ace  |  3  |      2      |      2       |      0
Dst…
```

   - **show pemain [namaTim] [nomorPemain atau namaPemain]**
     Perintah ini berfungsi untuk menampilkan informasi pemain. Output dari perintah **show pemain Bebek 13** atau **pemain Bebek Doffy** adalah sebagai berikut

```
Nomor : 13
Nama : Doffy
Gol : 15
Pelanggaran : 5
Kartu Kuning : 2
Kartu Merah : 1
```

- **`show nextGame`**
  Perintah ini berfungsi untuk menampilkan informasi pertandingan selanjutnya yang akan dilaksanakan. Perintah ini penting jika *user* ingin menggunakan model manual pada perintah **`nextGame`**. Dengan perintah ini, *user* akan mengetahui tim apa selanjutnya yang akan bertanding. **`show nextGame`** menghasilkan output sebagai berikut:

  ```
  Bebek VS Rajawali
  ```

Kerapihan presentasi pada fungsi **`show`** akan mendapat poin tambahan.

Jika Game telah usai, maka sistem akan menjalankan secara otomatis fungsi **`show klasemen`** dan **`show pencetakGol.`**

```
LIGA FASILKOM MUSIM INI TELAH USAI

CHAMPION: Ular

KLASEMEN

Peringkat | Nama Tim | Jumlah Gol | Jumlah Kebobolan | Menang | Kalah | Seri | Poin
-------------------------------------------------------------------------------------
    1     |   Ular   |     12     |         2        |   12   |   0   |   0  |  36
    2     |   Gajah  |     11     |         2        |   11   |   0   |   0  |  33

       Dst…


TOP SCORE

Peringkat | Nama Pemain | Nama Tim | Jumlah Gol
-----------------------------------------------
    1     |     Luffy   |  Bebek   |     31
    2     |     Zoro    |  Bebek   |     29
       Dst…


GOODBYE…
```

Poin Penilaian

1. Fungsi init (15%)
2. Fungsi nextGame (50%)
    a. Random (20%)
    b. Manual (20%)
    c. All (10)
3. Fungsi show (35%)
    a. Show klasemen (10%)
    b. Show pencetakGol (10%)
    c. Show tim (5)
    d. Show pemain (5)
    e. Show nextGame (5)
4. Error Handling (+5%)
5. Dokumentasi (+5%)