Network Laboratory
Electrical Engineering Department, 2nd floor
Universitas Indonesia
Depok, 16424

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK 2021/2022
CASE STUDY

Modul 7 – Exception, Threading, and Advance Logic

Referensi lib / dependencies:

java.io.*; java.lang.reflect.Array; java.util.Collections; java.util.Vector; com.google.gson.*;

A. JsonTable<T>

a. Buatlah sebuah class JsonTable dengan generic T yang inherit kepada Vector. Sesuaikan atribut class dengan yang diberikan pada UML.
b. Pastikanlah untuk instansiasi variabel gson secara static (jangan biarkan null).
c. Implementasikan method readJson untuk membaca file dengan format json yang ditentukan pada parameter. Class<T> merupakan referensi terhadap objek apa file json tersebut ingin di translasikan. contoh: readJson(Account.class, “account.json”); Syntax tersebut menandakan bahwa file account.json mengandung sebuah json object yang berisi atribut dari class Account. (ini memungkinkan juga untuk mereferensikan array type, contoh: Account[].class).
d. Implementasikan method writeJson untuk menulis suatu objek kepada filepath yang telah ditentukan pada parameter. Pastikan bahwa objek tersebut diubah kedalam bentuk Json sebelum ditulis ke storage.
e. Implementasikan instance method writeJson untuk menggunakan static writeJson dengan objek dan filepath mengarah kepada diri sendiri (JsonTable<T>).
f. Implementasikan constructor untuk melakukan assignment terhadap filepath dan kemudian membaca sebuah file yang ditentukan pada filepath. Objek yang ditranslasikan adalah bentuk array dari clazz yang diberikan pada parameter.

Setelah berhasil dimuat, jangan lupa untuk assign seluruh objek tersebut kepada diri kita sendiri (JsonTable<T> adalah Vector<T> adalah List<T>).

Perhatikan contoh berikut:

new JsonTable<>(Account.class, "account.json");

Instansiasi objek di atas menandakan bahwa file "account.json" **memuat** atau **digunakan untuk menyimpan** sebuah JSON Array dari JSON Object Account.

Berikut adalah corner cases yang perlu Anda atasi:

- Apabila file yang dimaksud tidak ada, maka Anda bertugas untuk membuat file tersebut. Ini menandakan bahwa konten masih kosong dan tidak menyebabkan kegagalan instansiasi. Artinya JsonTable diinisiasi tanpa elemen alias size 0.
- Apabila filepath mengandung tree directory seperti contoh: "a/b/c/account.json" maka Anda ditugaskan untuk **membuat** seluruh direktori yang dimaksud **apabila** direktori tersebut **tidak tersedia** pada saat instansiasi JsonTable. Apabila proses pembuatan direktori tersebut menghasilkan IOException, maka tidak perlu Anda handle, biarkan method melakukan throws.

g. Cek kebenaran code dengan contoh implementasi code dibawah.

```
try
{
    String filepath = "a/b/c/account.json";

    JsonTable<Account> tableAccount = new JsonTable<>(Account.class, filepath);
    tableAccount.add(new Account("name", "email", "password"));
    tableAccount.writeJson();

    tableAccount = new JsonTable<>(Account.class, filepath);
    tableAccount.forEach(account -> System.out.println(account.toString()));
}
catch (Throwable t)
{
    t.printStackTrace();
}
```

Run pertama, kondisi direktori dan file masih tidak ada:

```
name: name
email: email
password: password
```

Run kedua, kondisi direktori dan file sudah ada:

```
name: name
email: email
password: password
name: name
email: email
password: password
```

Kondisi File:

[{"name":"name","email":"email","password":"password","id":0},{"name":"name","email":"email","password":"password","id":0}]

Important Notes:

Penggunaan JsonTable<T> dan seluruh tipe Invoice akan digunakan pada saat proses Post Test. Diharapkan untuk menyelesaikan implementasi yang belum sempat diselesaikan dalam sesi Case Study. Poin pekerjaan pada bagian B tidak dinilai dalam Unit Test.

B. Penyesuaian Kode

a. (Tools untuk Modul Selanjutnya) Pada class Algorithm, deklarasi dan definisikan method paginate sesuai dengan yang diberikan pada UML. Silahkan merujuk kepada implementasi yang Anda lakukan pada modul sebelumnya!

b. Sesuaikan class Invoice dan atributnya sesuai dengan yang diberikan pada UML. Pastikan untuk inherit dari class Serializable pada package Anda! Assign nilai complaintId dengan -1, Date dengan tanggal saat ini, rating dengan Rating.NONE.

c. Sesuaikan class Payment dan atributnya sesuai dengan yang diberikan pada UML. Lakukan instansiasi ArrayList kosong pada instance variabel history. Pada constructor Payment.Record pastikan untuk instansiasi date dengan waktu saat ini.

d. Buatlah class PhoneTopUp yang inherit dari Invoice. Lakukan assignment sesuai dengan parameter yang diberikan pada constructor.

e. Lakukan implementasi pada method getTotalPay(Product) di setiap tipe Invoice. Gunakan logika Anda untuk mengembalikan hasil perhitungannya.