Dasar-Dasar Pemrograman 2

Tutorial 10 Kelas C dan F

Text I/O

Selasa, 30 April 2019 - 16:00 WIB

Materi
Pada TP2 kalian sebelumnya sudah ada *sneak-peek* mengenai materi lab kali ini, yaitu text I/O. Nah, kali ini akan diperjelas apa itu text I/O dan bagaimana cara menggunakannya.

Text I/O

Adalah cara untuk membaca Input dari Console atau file lain dan/atau mengeluarkan Output ke sebuah file lain yang pada umumnya ber-tipe data text (seperti .txt). Selain itu disini kita akan membahas bagaimana cara membuat suatu file dapat digunakan di dalam suatu program java.

Ada banyak *class* di java yang menyediakan fungsionalitas untuk *read/write* dari/ke file lain. Untuk sekarang ini, kita akan menggunakan *BufferedReader* untuk membaca, *FileReader* untuk membuat suatu file agar dapat dibaca oleh *BufferedReader*, *File* untuk membuat sebuah file dapat dimanipulasi dalam program dan *PrintWriter* untuk menulis output ke-sebuah file baru.

(Jika kalian kebingungan atau ingin mengetahui lebih banyak method - method dari kelas tersebut, kalian dapat selalu membuka Javadoc yang disediakan oleh java lho!)

Pembaca File

- **FileReader**
  FileReader adalah suatu class di Java yang bisa digunakan untuk membaca karakter dalam file.
  Contoh penggunaan:
  - Buat file dengan nama Hello1.txt berisi "**This is an example**"

  - Buat constructor class FileReader yang menerima parameter String nama file (dalam contoh ini "Hello1.txt") yang ada di direktori file Java (dalam contoh ini FileRead.java).

```
import java.io.*;
public class FileRead {

   Run | Debug
   public static void main(String args[])throws IOException {

        // Creates a FileReader Object
        FileReader fr = new FileReader("Hello1.txt");
```

- Kita bisa menyimpan seluruh karakter di file ke dalam array dengan method read(char[] c) dengan parameter array char tempat kita ingin menyimpan hasil file yang di-read.

```
char [] a = new char[50];
fr.read(a);   // reads the content to the array

for(char c : a)
    System.out.print(c);   // prints the characters one by one
fr.close();
```

- Output yang dihasilkan adalah seperti ini.

```
This
 is
 an
 example
```

- **BufferedReader**

  BufferedReader adalah suatu Class di Java yang berfungsi untuk menyederhanakan pembacaan teks dari suatu *input stream* (misalnya *file*). Secara singkat, BufferedReader meminimalisir penggunaan *I/O operation* dengan membaca potongan karakter dan menyimpannya di dalam *internal buffer* yang bisa menghasilkan pembacaan yang lebih cepat, karena tidak perlu berinteraksi lagi dengan *file* yang dibuka. BufferedReader biasanya digunakan dengan membaca suatu objek FileReader dan memasukkannya ke dalam *buffer*.

  Contoh penggunaan:

  - Buat terlebih dahulu file input.txt berisi "**Halo Dunia!**"
  - Selanjutnya buatlah program ReadFile.java berikut untuk membaca file input.txt

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ReadFile {
    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
        System.out.println(reader.readLine());
        reader.close();
    }
}
```

- Setelah dijalankan programnya akan menghasilkan sebagai berikut

```
dek.depe@cs.ui.ac.id:~$ ls
ReadFile.java input.txt
dek.depe@cs.ui.ac.id:~$ javac ReadFile.java
dek.depe@cs.ui.ac.id:~$ java ReadFile
Halo Dunia!
dek.depe@cs.ui.ac.id:~$
```

- Perhatikan di program ReadFile.java di atas, terdapat syntax 'throws IOException'. Seperti yang sudah dipelajari sebelumnya tentang materi *Error Handling*, pembacaan *file* input.txt dengan BufferedReader bisa saja terdapat *error* yaitu ketika *file* yang ingin dibaca tidak ditemukan.
-
- Perhatikan contoh keluaran berikut jika *file* input.txt tidak ditemukan.

```
dek.depe@cs.ui.ac.id:~$ ls
ReadFile.java
dek.depe@cs.ui.ac.id:~$ javac ReadFile.java
dek.depe@cs.ui.ac.id:~$ java ReadFile
Exception in thread "main" java.io.FileNotFoundException: input.txt (No such file or directory)
    at java.io.FileInputStream.open0(Native Method)
    at java.io.FileInputStream.open(FileInputStream.java:195)
    at java.io.FileInputStream.<init>(FileInputStream.java:138)
    at java.io.FileInputStream.<init>(FileInputStream.java:93)
    at java.io.FileReader.<init>(FileReader.java:58)
    at ReadFile.main(ReadFile.java:7)
dek.depe@cs.ui.ac.id:~$
```

- Kita juga bisa menggunakan 'try catch' *statement* saat menggunakan BufferedReader.

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ReadFile {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
            System.out.println(reader.readLine());
            reader.close();
        } catch (IOException e) {
            System.out.println("File tidak ditemukan");
        }
    }
}
```

- Lalu saat dijalankan dan *file* input.txt akan memberikan keluaran sebagai berikut.

```
dek.depe@cs.ui.ac.id:~$ ls
ReadFile.java
dek.depe@cs.ui.ac.id:~$ javac ReadFile.java
dek.depe@cs.ui.ac.id:~$ java ReadFile
File tidak ditemukan
dek.depe@cs.ui.ac.id:~$
```

- Kalian juga bisa mencoba bagaimana jika program ReadFile.java tersebut tidak menggunakan ‘throws IOException’ ataupun ‘try catch’ *statement*.

- **File**

File adalah suatu class di java yang merepresentasikan suatu “file” dalam program java. Objek dari kelas File dapat menyimpan berbagai atribut dari “file” dalam komputer itu sendiri seperti path dan sifatnya (apakah hanya bisa dibaca, apakah hanya bisa di-*write* dan sebagainya). Sekali dibuat, suatu *instance* dari class file tidak dapat diubah pathnya.

Contoh penggunaan :

```java
import java.io.File;

class fileboi{
    Run | Debug
    public static void main(String[]args) {
        File f = new File("ini.txt"); //membuka file bernama ini.txt

        System.out.println(f.getName()); //ini.txt
        System.out.println(f.getPath()); //ini.txt
        System.out.println(f.getAbsolutePath()); //C:\Users\arnas\Desktop\Temp\ini.txt
    }
}
```

Untuk info tentang *method-method* yang ada dalam class File silahkan merujuk kepada tautan berikut ini.

Penulis File

- **PrintWriter**
  Printwriter adalah sebuah class yang membantu program java untuk mencetak secara langsung kedalam class File yang telah dibuat. Perbedaan Printwriter dengan beberapa class untuk mencetak lainnya adalah Printwriter ditujukan untuk mencetak sebuah "teks" kedalam file, oleh karena itu PrintWriter dilengkapi dengan method format() dan printf(). PrintWriter mengimplementasikan method yang ada dalam PrintStream, oleh karena itu PrintWriter **harus di close()** ketika sudah selesai penggunaannya. Selengkapnya pada tautan berikut.

  **Perhatikan bahwa PrintWriter dapat memicu adanya FileNotFoundException pada pembuatan *instance* nya. Kira-kira bagaimana cara meng*handle* nya ya ? :)

Contoh penggunaan :

```java
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;

class fileboi{
    Run | Debug
    public static void main(String[]args) throws IOException {
        File f = new File("ini.txt"); //membuka file bernama ini.txt

        System.out.println(f.getName()); //ini.txt
        System.out.println(f.getPath()); //ini.txt
        System.out.println(f.getAbsolutePath()); //C:\Users\arnas\Desktop\Temp\ini.txt

        PrintWriter pw = new PrintWriter(f);
        pw.println("hai dunia");
        pw.printf("%d", 99);
        pw.close();
    }
}
```

Breakfast Menu

Pada suatu hari, Dek Depe diminta untuk berkolaborasi dengan Bang Basdat untuk membuat Menu makanan di restoran barunya. Bang Basdat membutuhkan query SQL untuk memasukkan menu-menu yang diberikan chef nya dalam bentuk CSV. Oleh sebab itu, Dek Depe minta bantuan kamu untuk membuatkan converter dari CSV menjadi sintaks insert query SQL. Tenang saja, kamu tidak perlu untuk mempelajari SQL kok, paling tidak belum sekarang..

Kamu diminta untuk membuat program yang mensimulasikan proses pengolahan data CSV menjadi sebuah query statement dalam konteks SQL.

Comma separated values, atau biasa disebut CSV, adalah sebuah format *file* sederhana yang biasa digunakan untuk menyimpan data yang tersusun dalam beberapa baris, seperti sebuah spreadsheet atau *database*. Satu baris dalam file CSV adalah sebuah *record*, di mana satu *record* dapat memiliki banyak *data fields* yang dibatasi (delimited) dengan tanda koma. Contohnya adalah sebagai berikut:

```
no ktp,no_rekening,kantor_cabang,nama_bank
3274212597827940,0907984212,Jakarta Selatan,Bank BNI
3278582361758580,111991018832,Jakarta Selatan,Bank Mandiri
3274692400892570,0656992156,Jakarta Selatan,Bank BNI
3275954822239780,0681171621,Cimahi,Bank BNI
3271786541831160,0656831317,Jakarta Selatan,Bank BNI
3274860243830410,111988744124,Jakarta Timur,Bank Mandiri
```

Juga terdapat satu baris header yang ada di baris pertama *file* CSV dengan format yang sama seperti baris *record*. *Header* berisi nama yang sesuai dengan *data field* dari *record* yang ada di file.

Spesifikasi Program :

Kamu diminta untuk mengubah sebuah CSV file menjadi kumpulan *insert query* PostGreSQL dalam sebuah file .txt, serta method filter berdasarkan field yang ada. Kamu akan diberikan template yang terdiri dari 3 class sebagai berikut:

1. **Item**
2. **Menu**
3. **MenuParser**

Di dalam ketiga class yang disediakan terdapat method yang belum memiliki implementasi, **silahkan implementasikan method sesuai dengan ketentuan yang ada di template**. Kamu juga diberikan sebuah Main class yang akan di-*run*, **pastikan kode kalian memenuhi kriteria *file* ini**.

Berikut adalah contoh output dijalankan jika diberikan input.csv.

Input:

[input.csv]

```
id,name,price,type
1,FOOD NAME 1,9.72,MINERAL
2,FOOD NAME 2,6.22,WATER
3,FOOD NAME 3,8.08,FAT
4,FOOD NAME 4,2.6,MINERAL
5,FOOD NAME 5,9.59,FAT
6,FOOD NAME 6,9.6,CARBOHYDRATE
7,FOOD NAME 7,7.2,WATER
8,FOOD NAME 8,2.58,FIBRE
9,FOOD NAME 9,5.33,VITAMIN
10,FOOD NAME 10,6.62,PROTEIN
```

dan seterusnya..

Output:

all.txt

```
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (1, FOOD NAME 1, 9.72, MINERAL);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (2, FOOD NAME 2, 6.22, WATER);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (3, FOOD NAME 3, 8.08, FAT);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (4, FOOD NAME 4, 2.6, MINERAL);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (5, FOOD NAME 5, 9.59, FAT);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (6, FOOD NAME 6, 9.6, CARBOHYDRATE);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (7, FOOD NAME 7, 7.2, WATER);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (8, FOOD NAME 8, 2.58, FIBRE);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (9, FOOD NAME 9, 5.33, VITAMIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (10, FOOD NAME 10, 6.62, PROTEIN);
```

dan seterusnya..

filterPrice.txt

```
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (4, FOOD NAME 4, 2.6, MINERAL);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (8, FOOD NAME 8, 2.58, FIBRE);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (11, FOOD NAME 11, 2.48, MINERAL);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (13, FOOD NAME 13, 3.35, FAT);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (19, FOOD NAME 19, 3.56, MINERAL);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (21, FOOD NAME 21, 4.2, MINERAL);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (22, FOOD NAME 22, 1.02, VITAMIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (23, FOOD NAME 23, 1.32, CARBOHYDRATE);
```

dan seterusnya..

filterType.txt

```
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (10, FOOD NAME 10, 6.62, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (15, FOOD NAME 15, 7.42, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (26, FOOD NAME 26, 4.4, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (38, FOOD NAME 38, 2.3, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (40, FOOD NAME 40, 6.91, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (49, FOOD NAME 49, 7.92, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (54, FOOD NAME 54, 3.67, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (55, FOOD NAME 55, 4.92, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (62, FOOD NAME 62, 6.95, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (66, FOOD NAME 66, 8.29, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (74, FOOD NAME 74, 4.37, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (78, FOOD NAME 78, 5.48, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (85, FOOD NAME 85, 1.74, PROTEIN);
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (100, FOOD NAME 100, 7.62, PROTEIN);
```

getId.txt

```
INSERT INTO MENU (ID, NAME, PRICE, TYPE) VALUES (19, FOOD NAME 19, 3.56, MINERAL);
```

Versi lengkap contoh input dan output juga diberikan bersamaan dengan template, silahkan sesuaikan output program anda sesuai dengan contoh output yang sudah diberikan.

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Implementasi Class Item	Implementasi pemenuhan class Item sesuai dengan kebutuhan soal.	10 %
Implementasi Class Menu	Implementasi pemenuhan method getInsertQuery() dan semua jenis method filter dalam class Menu.	30 %
Implementasi Class MenuParser	Implementasi pemenuhan method parseMenu() dalam class MenuParser untuk parsing CSV file ke format Menu.	15 %

Implementasi Class Main	Implementasi pemenuhan method writeToFile() untuk dalam class Main untuk menampilkan output dalam file .txt	15 %
Ketepatan Output	Ketepatan Output ketika Main dijalankan (Pemenuhan requirement client).	20 %
Kerapian	Penulisan program mengikuti kaidah dan konvensi yang telah diajarkan. Program ditulis dengan rapi, terstruktur, dan disertakan oleh dokumentasi secukupnya.	10 %

Deadline :

Selasa, 30 April 2019
Pukul **17:40 WIB**

Persentase Nilai Total Terhadap Lama Keterlambatan :

dari	hingga	persentase
0:00:00	0:00:00	100%
0:00:01	0:10:00	85%
0:10:01	0:20:00	70%
0:20:01	0:30:00	50%
0:30:01	24:00:00	25%

Format Pengumpulan :

Zip semua class yang kamu buat dan kumpulkan di slot pengumpulan yang telah disediakan di SCeLE dengan format :

[Kode Asdos]_[Nama]_[Kelas]_[NPM]_Lab[X].zip

Contoh :

JO_JonathanChristopherJakub_C_1706040151_Lab10.zip

Acknowledged Lecturers :

- Fariz Darari, S.Kom, M.Sc., Ph.D.

Authors :

- JO
- SAM
- KC
- DN