Modul Prak. Struktur Data 7 - Java Collection Part 1

INSTITUT TEKNOLOGI HARAPAN BANGSA
Veritas vos liberabit

Java sebagai bahasa pemrograman modern, telah menyediakan berbagai Collections atau struktur data yang telah siap dipakai. Lalu mengapa kita perlu mempelajari cara kerja detailnya di kelas Teori, bukankah tinggal dipakai saja? Karena sebagai calon Sarjana Informatika, inilah yang membedekan saudara dengan lulusan SMK ataupun D3. Kalau hanya programming biasa, mereka pun hebat-hebat. Namun bagi sarjana, kalian dituntut memiliki pemahaman yang lebih dalam, sehingga dapat menyelesaikan masalah yang kompleks dan juga mampu membuat struktur data sendiri yang belum disediakan oleh bahasa pemrograman.

Namun, dalam pembuatan software pada umumnya, kalian justru disarankan menggunakan Collections, karena untuk apa membuat lagi sendiri hal yang sudah umum? Serta setiap Collections ini juga telah dioptimalisasi performansinya, belum tentu ketika kita membuat sendiri bisa lebih efisien daripada yang telah dibuat oleh Java. Hari ini akan diberikan beberapa contoh Collections yang paling sering digunakan beserta latihan soal di akhir.

NOTES
1. Masih terdapat banyak fungsi yang dapat digunakan untuk masing-masing Collections seperti RemoveAll, addAll, dan lain sebagainya. Anda dapat melihatnya di internet, contoh : https://www.w3schools.com/java/java_arraylist.asp
2. Ada beberapa cara untuk melakukan print data, remove, dll. Tidak harus persis seperti yang dicontohkan di atas.
3. Contoh yang diberikan mungkin saja akan error apabila anda langsung copy ke IDE dari pdf, terutama untuk spasi dan tanda kutip yang berbeda. Silahkan disesuaikan lagi sendiri, saran saya sebaiknya diketik ulang saja.
4. Jangan lupa anda perlu meng-import Collections tersebut. Caranya setelah anda ketik misal LinkedList<String> maka pasti akan ada alert merah di pinggir, klik dan pilih Add Import sehingga akan muncul tulisan seperti "import java.util.LinkedList" di bagian atas file.

1. LinkedList

LinkedList sama seperti yang dipelajari di kelas Teori, dan implementasinya di dalam Java menggunakan Doubly Linked List.

```
LinkedList<String> list = new LinkedList<String>();
list.add("Budi"); //insert Budi at last : {Budi}
list.add("Heru"); //insert Heru at last : {Budi, Heru}
list.add(1, "Andi"); //insert Andi at position 1 : {Budi, Andi, Heru}

// Untuk print harus menggunakan Iterator dan fungsi next() karena list tidak bisa langsung
di-get berdasarkan indexnya. Sehingga di-get dengan cara looping satu-per-satu (next).
Iterator iterator = list.iterator();
while(iterator.hasNext()) {
        System.out.println(iterator.next()); //looping print Budi, Heru, Andi

        // Apabila butuh untuk mendapatkan value-nya terlebih dahulu tanpa di-print,
        // anda dapat menggunakan fungsi casting, perhatikan tanda "(String)" di depan
        // "iterator.next()" berguna untuk men-define bahwa tipe datanya adalah String.
        String temp = (String) iterator.next();
}
```

```
// Untuk menghapus dapat menggunakan value isinya (dalam hal ini String nama), atau fungsi removeFirst() dan removeLast(). List tidak memiliki informasi index-nya.
list.remove("Andi"); //remove Andi
list.removeLast(); //remove Heru
```

2. ArrayList

Bayangkan ArrayList sebagai penggabungan dari Array dan List yang memiliki karakteristik :

- Ukuran / size-nya unlimited dan tidak perlu di-define seperti LinkedList
- Dapat diambil langsung menggunakan index seperti Array.

Collection ini adalah ciri khas Java yang belum tentu akan anda temukan di dalam bahasa pemrograman lainnya (sebagian bahasa pemrograman modern mempunyai padanan yang sama, tetapi tidak semua).

```
ArrayList<String> arrayList = new ArrayList<String>();
arrayList.add("Budi");
arrayList.add("Heru");

// Untuk print dapat dilakukan looping for biasa, karena ArrayList dapat di-get dengan index.
System.out.println(arrayList.get(1)); //print Heru
for(int i=0; i<arrayList.size(); i++) {
        System.out.println(arrayList.get(i)); //looping print Budi and Heru

        // Apabila butuh value-nya saja dapat juga disimpan dalam variabel berikut.
        String temp = arrayList.get(i);
}

// Untuk menghapus dapat langsung menggunakan Index-nya
arrayList.remove(1); //remove Heru
```

3. HashSet

HashSet adalah implementasi Java untuk Set, seperti yang sudah pernah diajarkan di kelas Teori. Set ini hanya dapat menyimpan data value dan tidak memiliki Index atau urutan. Data yang disimpan harus *unique* atau tidak bisa duplikat.

```
HashSet<String> set = new HashSet<String>();
set.add("Budi");
set.add("Heru");

// Untuk print dapat dilakukan looping for biasa, karena ArrayList dapat di-get dengan index.
Iterator iterator = set.iterator();
while(set.hasNext()) {
        System.out.println(set.next()); //looping print Budi, Heru
        // Apabila butuh value-nya saja dapat juga disimpan dalam variabel sementara
        // Implementasinya sama seperti pada LinkedList.
}
```

```
// Untuk menghapus harus menggunakan value-nya, karena set tidak memiliki Index.
set.remove("Budi"); //remove Budi
```

4. HashMap

HashMap adalah implementasi Java untuk Dictionary/Map, seperti yang sudah pernah diajarkan di kelas Teori.

```
HashMap<Integer, String> map = new HashMap<Integer, String>();
map.put(101, "Budi");
map.put(102, "Heru");

// Mengganti value yang sudah pernah di-insert
map.replace(102, "Andi");

// Untuk mendapatkan value kita perlu get by key
System.out.println(map.get(102)); //print Andi

// Untuk print dapat dilakukan looping untuk setiap key menggunakan fungsi keySet(),
kemudian value-nya diget dengan fungsi get(key) biasa.
for(Integer i : map.keySet()) {
        System.out.println("key : " + i + " value : " + map.get(i));
}

// Untuk menghapus dapat menggunakan key-nya
arrayList.remove(102); //remove Andi
```

5. Collection with Class

Setiap collection juga dapat digabungkan penggunaannya dengan tipe data bentukan (struct/class/type/record). Perhatikan contoh berikut apabila di-asumsikan kita memiliki suatu Class Person.

```
class Person {
        String nama;
        int umur;
}
// contoh implementasi HashMap dengan Struct
HashMap<String, Person> map = new HashMap<String, Person>()
Member m1 = new Member(nama, umur);
map.put("123", m1);

// contoh implementasi HashMap dengan
Member m = map.get("123");
System.our.println(m.nama + " " + m.umur);

// Implementasi untuk LinkedList, ArrayList, dan HashSet juga sama saja.
```

LATIHAN

Buatlah sebuah program sederhana untuk perpustakaan dengan ketentuan memiliki struktur data seperti berikut :

```
class Member {
        String nama;
        String telepon;

        // constructor yang sesuai
}
```

Data member disimpan dalam sebuah **HashMap** dengan **Key**: IDMember (String) dan **Value**: Member.

```
HashMap<String, Member> map = new HashMap<String, Member>()
```

```
class Buku {
        String idBuku;
        String judul;
}
```

Data buku disimpan dalam sebuah **Linked List.**

```
LinkedList<Buku> list = new LinkedList<Buku>()
```

```
class Peminjaman {
        String idMember;
        String idBuku;
}
```

Data Peminjaman disimpan dalam sebuah **ArrayList**.

```
ArrayList<Peminjaman> arrayList = new ArrayList<Peminjaman>()
```

Program Perpusatakaan ini memiliki beberapa fitur :

1. Tambah baru dan hapus data member (dari/ke dalam HashMap). Hapus berdasarkan input nama.
2. Tambah baru dan hapus data buku (dari/ke dalam Linked List). Hapus berdasarkan input IDBuku.
3. Melakukan peminjaman. Program meminta input IDMember dan IDBuku. Lalu ditambahkan data peminjaman baru ke dalam ArrayList Peminjaman. Lakukan pengecekan terlebih dahulu apakah IDMember dan IDBuku yang di-input tersedia di dalam HashMap dan LinkedList.
4. Selesai meminjam. Program meminta input IDMember dan IDBuku, lalu hapus data dari ArrayList Peminjaman.

Bonus :

Anda dapat mengembangkan sebuah fitur baru yang memanfaatkan HashSet, seperti contoh untuk menyimpan nomor undian, dan lain sebagainya. Hanya kerjakan soal bonus ini apabila anda masih memiliki waktu tersisa.

Ketentuan :

1. Buatlah cukup 2 data *dummy* atau data awal Member, Buku, dan Peminjaman yang sudah diisikan ke dalam Colllections. Supaya anda tidak perlu input dulu setiap kali di-run.
2. Anda dapat menggunakan Scanner atau JOptionPane.
3. Penekanan pada implementasi Collections. Pembuatan menu, tampilan, pesan error, dll. apabila masih ada sisa waktu saja.