Modul Prak. Struktur Data 8 - Java Collection Part 2

Dalam pembahasan modul sebelumnya, telah dibahas collection ArrayList, LinkedList, HashMap, dan HashSet. Pada modul ini, akan dibahas collection lainnya yakni Stack dan Queue.

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. Stack

Implementasi Stack di Java sama seperti yang telah kita pelajari di kelas teori.

```
// deklarasi Stack
Stack<Integer> stack = new Stack<Integer>();

// mengisi data ke dalam stack
stack.push(5);
stack.push(13);

// mencari posisi dari suatu value di dalam stack
Integer pos = (Integer) stack.search(5);
if(pos == -1) {
   System.out.println("Element not found");
} else {
   System.out.println("Element is found");
}

// melihat nilai value teratas (tidak dihapus)
Integer peekValue = (Integer) stack.peek();
System.out.println(peekValue);  // print 13, stack: 5,13

// mengeluarkan value teratas, lalu dihapus
Integer popValue = (Integer) stack.pop();
System.out.println(popValue);   // print 13, stack : 5
```

2. Queue : LinkedList

Implementasi Queue di Java dapat menggunakan 2 bentuk yakni LinkedList dan PriorityQueue. LinkedList digunakan karena sifatnya yang dinamis dibandingkan array dan tidak perlu mengurus proses pergeseran / sirkular. Namun bagaimana jika kita membutuhkan implementasi Queue yang menggunakan array, maka anda harus mengimplementasikan sendiri seperti tugas Dequeue yang telah anda kerjakan.

```
// deklarasi Queue LinkedList
Queue<Integer> queue = new LinkedList<Integer>();

// mengisi data ke dalam queue
queue.add(5);
queue.add(13);

// melihat ukuran queue
System.out.println(queue.size()); // print 2

// melihat nilai value terdepan (tidak dihapus)
Integer peekValue = (Integer) queue.peek();
System.out.println(peekValue);  // print 5, queue: 5,13

// mengeluarkan value terdepan, lalu dihapus
Integer value = (Integer) queue.remove();
System.out.println(value);   // print 5, queue : 13
```

3. Queue : Priority Queue

PriorityQueue lebih kurang sama saja dengan implementasi Queue LinkedList, hanya ketika proses dequeue (pengambilan data, akan dilakukan pencarian terlebih dahulu data mana yang lebih tinggi prioritasnya, ini akan dibahas lebih lanjut di kelas Teori), namun karena ini pertemuan praktikum terakhir, maka dibahas juga di dalam modul ini. Apabila tipe data pada Queue adalah Integer, maka defaultnya akan diurutkan secara ascending, apabila tipe datanya Char/String maka diurutkan berdasarkan ASCII nya. Sedangkan apabila tipe datanya adalah struct/tipe data buatan anda sendiri, maka anda perlu membuat metode komparasi sendiri (tidak dijelaskan di sini).

```
// deklarasi Queue PriorityQueue
Queue<IntSeger> queue = new PriorityQueue<Integer>();

// mengisi data ke dalam stack
queue.add(5);
queue.add(13);
queue.add(7);

// melihat nilai value terdepan (tidak dihapus)
Integer peekValue = (Integer) queue.peek();
System.out.println(peekValue);  // print 5, queue: 5,7,13 (diurut ascending)
```

```
// mengeluarkan value terdepan, lalu dihapus
Integer value = (Integer) queue.poll();
System.out.println(value);   // print 5, queue : 7,13
```

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;
    Queue<String> waitingList = new LinkedList<String>() // untuk menyimpan IDMember
}

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>()
```

Modifikasilah program perpustakaan yang telah anda kembangkan pada praktikum sebelumnya.

1. **Modifikasi fitur Tambah Buku**, yang sebelumnya hanya langsung input IDBuku dan Judulnya lalu disimpan dalam LinkedList, menjadi 2 tahapan :
    1. **Beli Buku**. Anggaplah buku baru dibeli oleh perpustakaan, akan disimpan dulu dalam sebuah Stack (ditumpuk), input hanya judul buku-nya saja, tanpa ID-nya.
    2. **Registrasi Buku**. Buku baru yang sudah ditumpuk, kemudian harus di-Registrasikan, dimana ditambahkan data ID (input), barulah data Buku yang sudah lengkap ini (ID dan Judul) dimasukkan ke dalam LinkedList seperti sebelumnya,
2. **Modifikasi fitur Peminjaman Buku**, dengan fitur Waiting List.
    1. Pada latihan sebelumnya, saat peminjaman hanya perlu input IDMember dan IDBuku, lalu disimpan dalam ArrayList.
    2. Namun bagaimana bila IDBuku tersebut sebenernya sedang dipinjam, maka anda dapat memasukkan IDMember yang ingin meminjam ini ke dalam sebuah Queue.

3. Saat pengembalian buku (data peminjaman dihapus dari ArrayList), langsung dicek apakah ada antrian dalam Queue, apabila ada maka otomatis masuk lagi peminjaman baru dengan IDMember tersebut ke dalam ArrayList Peminjaman.
4. Dapat dilihat pada Struktur Data **Buku** yang diberikan, terdapat tambahan variabel **waitingList** berupa Queue LinkedList yang digunakan untuk menyimpan IDMember yang mengantri. Dengan demikian bisa dikatakan untuk setiap buku, terdapat antrian-nya masing-masing. Queue yang digunakan adalah LinkedList, tidak perlu Priority.

Ketentuan :
1. Anda dapat melakukan penyesuaian struktur data Class yang digunakan apabila diperlukan, sebagai contoh ID yang String dapat anda ubah menjadi Integer bila dirasa lebih mudah.
2. Apabila ada yang kurang jelas, anda dapat menggunakan asumsi / variasi anda sendiri selama tidak mengubah fiturnya.