IMPLEMENTASI STRUKTUR DATA LIST, QUEUE DAN STACK DALAM JAVA

Rachmat Selamet

Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI
Jl. Ir. H. Juanda 96 Bandung 40132

E-mail : rachmatselametskom@gmail.com

Abstrak

Java adalah pemograman yang menggunakan konsep *object oriented* murni. Dengan menggunakan versi JDK (*Java Develompent Kit*) dari Java, maka kelas untuk mengoperasikan *list*, *queue* dan *stack* sudah tersedia. Untuk mengakses data dari kelas-kelas tersebut dapat dikonversi menjadi *array* atau *iterator* (list satu arah).

Dalam Java, List dapat diimplementasikan menggunakan LinkedList atau ArrayList atau Vector sesuai dengan fungsinya. LinkedList digunakan untuk menyimpan data yang sering ditambah atau dihapus. ArrayList digunakan untuk menyimpan data yang sering dicari atau diakses. Vector digunakan untuk menyimpan data yang digunakan oleh lebih dari 1 proses. Queue adalah struktur yang memproses data secara FIFO (*First In First Out*) dan Stack adalah struktur yang memproses data secara LIFO (*Last In First Out*). Java sudah menyedikan LinkedList yang digunakan sebagai queue, Stack yang digunakan sebagai stack, dan ArrayDeque sebagai deque (gabungan antara queue dan stack). PriorityQueue digunakan untuk kasus yang memiliki kriteria khusus untuk didahulukan.

Kata-kata kunci : *Java, list, queue, stack*

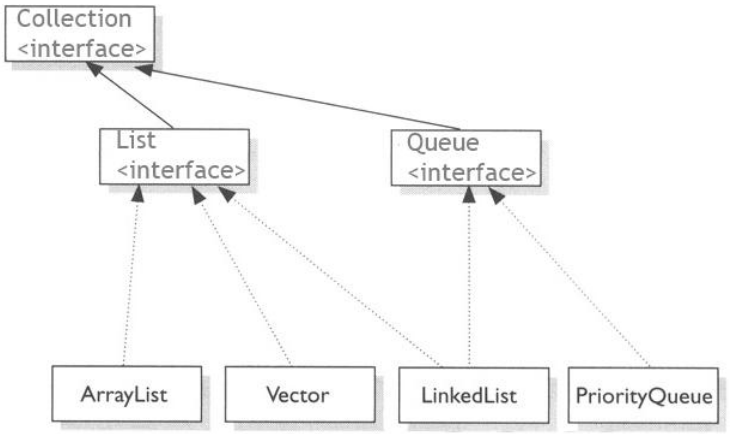
1 PENDAHULUAN

Dunia pemograman mengalami kemajuan pesat. Hal ini ditandai dengan makin berkurangnya penggunaan program secara prosedural dan makin bertambahnya penggunaan konsep *object oriented*. Salah satu bahasa pemograman yang murni menggunakan konsep *object oriented* adalah Java.

Untuk kebutuhan pemograman, versi Java yang digunakan adalah JDK (*Java Development Kit*). Dalam versi ini, Java sudah menyediakan kebutuhan umum dalam implementasi struktur data secara umum, seperti *list*, *queue* dan *stack* dalam bentuk

kelas di Java. Kelas-kelas tersebut memiliki atribut dan *method* yang menggunakan istilah yang sama dalam dunia struktur data.

Dalam paper ini akan dibahas bagaimana cara menggunakan kelas yang berhubungan dengan struktur data. Khusus untuk mengakses data, kelas-kelas ini secara umum dapat dikonversi menjadi *array* atau *iterator* (list satu arah). Hal ini disebabkan karena list dan queue diturunkan atau diimplementasi dari kelas Collection (dapat dilihat pada gambar 1). Kelas inilah yang memiliki operasi untuk konversi *array* dan *iterator*.



Gambar 1. Hirarki kelas untuk kebutuhan struktur data [3]

2 ARRAY DAN ITERATOR

Array adalah struktur data yang paling umum digunakan dalam semua bahasa pemograman. Deklarasi array dalam Java menggunakan kurung kotak '[ ]' dengan isi jumlah maksimal yang dapat ditampung oleh array. Sedangkan index yang dapat diakses oleh array adalah dari 0 sampai jumlah maksimal-1. Contoh akan diberikan pada bab 3 tentang mengakses data menggunakan *array*.

Iterator adalah sebuah list 1 arah yang berfungsi untuk mengakses semua data dari list. Kelas ini hanya memiliki 2 *method*, yaitu hasNext() (untuk memeriksa apakah masih ada data selanjutnya atau tidak) dan next() (untuk mengambil data). Contoh akan diberikan pada bagian tentang mengakses data menggunakan *iterator*.

3 LIST

List adalah sekumpulan data sejenis yang berada dalam lokasi yang fleksibel (dapat berubah sesuai dengan kebutuhan). Java sudah menyediakan 3 jenis kelas yang berhubungan dengan list, yaitu :

a. **LinkedList** : kelas yang berisi dobel linked list (data dapat diakses dari depan atau belakang)
b. **ArrayList** : kelas yang mengimpelemtasikan array tetapi dapat beroperasi seperti list (dapat bertambah atau berkurang otomatis)
c. **Vector** : kelas yang mirip dengan arraylist tetapi memiliki thread-safe, artinya aman digunakan jika diakses lebih dari 1 proses.

Perbedaan antara LinkedList dengan ArrayList :

a. **Pencarian data** : ArrayList melakukan pencarian jauh lebih cepat karena menggunakan index (seperti array) sedangkan LinkedList mencari per-elemen
b. **Menghapus data** : LinkedList lebih cepat terutama dalam menghapus elemen pertama atau terakhir (tinggal menggeser pointer) dibandingkan ArrayList (karena harus menggeser posisi index elemen/item dari array untuk menghapus elemen)
c. **Menyisipkan data** : LinkedList lebih baik dari ArrayList dengan alasan sama dengan menghapus.
d. **Penggunaan memori** : ArrayList menggunakan index untuk pemilihan data sedangkan LinkedList menggunakan pointer untuk data tetangganya yang memakan memori jauh lebih besar dibandingkan index dari Array

Perbedaan antara ArrayList dengan Vector :

a. ***Synchronization*** : ArrayList *non-synchronized* artinya hanya dapat dikerjakan 1 proses dalam 1 waktu. Sedangkan Vector *synchronized*.
b. ***Resize*** : Kedua kelas dapat bertumbuh dan mengecil secara dinamis tetapi ArrayList memakan ukuran setengah dari ukurannya sedangkan Vector memakan 2x dari ukurannya
c. ***Performance*** : ArrayList lebih baik performanya karena non-synchronized, sedangkan Vector harus mengurusi dahulu thread-safe (bagaimana mengatur penguncian proses apabila ada proses lain datang)
d. ***Fail-fast*** : Vector tidak fail-fast, sedangkan ArrayList fail-fast. Fail-fast adalah bagaimana program mendukung reaksi cepat struktur data dalam mengalami perubahan.

Pembuatan program menggunakan List:

a. **Deklarasi** :

JenisList<data> namavar = new JenisList<data>();

Contoh : ArrayList<String> isidata=new ArrayList<String>();

b. **Menambah data** :

namavar.add(nilai data);

Contoh : isidata.add("Budiman");

c. **Menghapus data** :

namavar.remove(nilai data);

Contoh : isidata.remove("Budiman");

d. **Mengubah data** :

namavar.set(nilai lama, nilai baru);

Contoh : isidata.set("Budiman","Anton"):

e. **Memeriksa kosong** :

namavar.isEmpty();

Contoh : if (isidata.isEmpty()) System.out.println("Isi list kosong!");

f. **Memeriksa data** :

namavar.contains(nilai data);

Contoh : if (isidata.contains("Anton")) System.out.println("Anton ada!");

g. **Mengitung posisi data data** :

namavar.indexOf(nilai data);

Contoh : if (isidata.contains("Anton"))
System.out.printf("%d\n", isidata.indexOf ("Anton"));

h. **Mengitung jumlah data** :

namavar.size();

Contoh : System.out.printf("%d\n",isidata.size());

i. **Mengakses data menggunakan iterator** :

```
Iterator <String> itr=isidata.iterator();
String skrg = "";
while (itr.hasNext()){
   skrg = itr.next();
   System.out.println(skrg);
}
```

j. **Mengakses data menggunakan array** :

```
String dt[]=new String[isidata.size()];
isidata.toArray(dt);
```

```
for (int i=0;i<dt.length;i++){
    System.out.println(dt[i]);
}
```

4 QUEUE DAN STACK

Queue adalah struktur data yang mengakses data secara FIFO (yang awal datang, yang awal diproses). Kelas di Java yang dapat mengimplementasikan queue adalah LinkedList dan PriorityQueue.

Stack adalah struktur data yang mengakses data secara LIFO (yang terakhir datang, yang awal diproses). Kelas di Java yang dapat mengimplementasikan stack adalah Stack. Stack merupakan turunan dari Vector, sehingga seluruh operasi dari vector dapat digunakan di Stack.

Deque adalah struktur data gabungan dari queue dan stack. Deque memiliki seluruh operasi queue dan seluruh operasi stack. Kelas di Java yang dapat mengimpelemtasikan deque adalah ArrayDeque.

Pembuatan program untuk queue dan stack :

a. **Deklarasi** : :

   QueueStack<data> namavar = new QueueStack<data>();

   Contoh : Stack<String> isidata = new Stack<String>();

b. **Menambah data** :

   Queue.offer(nilai data);

   Stack.push(nilai data);

   Contoh : isidata.push(“Budiman”);

c. **Mengambil data pertama dan menghapusnya** :

   Queue.poll();

   Stack.pop();

   Contoh : String skrg=isidata.pop();

d. **Mengambil data pertama dan menghapusnya** :

   namavar.peek();

   Contoh : String skrg=isidata.peek();

PrirortyQueue adalah jenis queue yang dapat memberikan prioritas untuk akses data yang lain. Contoh :

```
public class ContohPriorityQueue {
  public static void main(String[] args) {
    public static PriorityQueue<Pasien> qpasien = new PriorityQueue<Pasien>(10,
    new Comparator<Pasien>() {
      public int compare(Pasien pasien1, Pasien pasien2) {
        return (pasien1.isdarurat() == pasien2.isdarurat()) ?
              (Integer.valueOf(pasien1.nomor).compareTo(pasien2.nomor)) :
              (pasien1.darurat ? -1 : 1);
      }
    }
   );
qpasien.offer(new Pasien(1,"Anton",false));
qpasien.offer(new Pasien(2,"Budi",true));
qpasien.offer(new Pasien(3,"Bambang",false));
qpasien.offer(new Pasien(4,"Andi",true));
skrg = dt.poll();
while (skrg != null) {
        System.out.print(skrg.getNomor() + " " + skrg.getNama() + " " +
              skrg.isDarurat() + " <-- ");
        skrg = dt.poll();}
        System.out.println();    }
public static class Pasien{
        private int nomor;
        private String nama;
        private boolean darurat;
        public boolean isDarurat() {
              return darurat;
        }
        public void setDarurat(boolean darurat) {
              this.darurat = darurat;
        }
        public String getNama() {
              return nama;
        }
        public void setNama(String nama) {
              this.nama = nama;
        }
        public int getNomor() {
              return nomor;
        }
        public void setNomor(int nomor) {
              this.nomor = nomor;
        }
        public Pasien(int id, String n,boolean d){
              nomor = id; nama = n; darurat = d;
        }
   }
}
```

Pada contoh program PriortyQueue pasien, data masuk berdasarkan urutan input data. Pada saat data diambil menggunakan poll, maka data akan diambil otomatis berdasarkan comparator. Pada contoh pasien, comparator yang digunakan adalah untuk membandingkan apakah pasien tersebut darurat atau tidak, jika darurat akan didahulukan dibandingkan berdasarkan nomor.

5 KESIMPULAN

Kesimpulan yang dapat ditarik dari hasil perancangan dan implementasi perangkat lunak permainan musik digital ini adalah sebagai berikut :

a. Dengan menggunakan Java versi JDK, implementasi struktur data menjadi lebih mudah karena sudah disediakan dalam bentuk kelas dengan istilah-istilah yang sama dengan yang digunakan dalam struktur data.
b. Semua struktur list, queue dan stack dapat diakses dengan mengkonversi menjadi bentuk array atau iterator.
c. LinkedList secara umum digunakan untuk kasus yang banyak berhubungan dengan insert dan delete, misalnya : kasus saham dan valuta asing.
d. ArrayList secara umum digunakan untuk kasus yang banyak berhubungan dengan pengaksesan data secara cepat. Tipe data ini yang paling banyak digunakan untuk kasus-kasus database di perusahaan.
e. Vector secara umum digunakan untuk kasus yang berhubungan dengan pengaksesan data secara cepat tetapi digunakan untuk lebih dari 1 proses. Tipe data ini digunakan untuk kasus-kasus database yang datanya didapat diakses oleh beberapa pemakai (*multi user*).
f. Queue merupakan struktur yang paling banyak digunakan untuk pemrosesan data karena pengaksesan data dimulai dari awal.
g. Stack digunakan untuk kasus yang memerlukan pemrosesan data dari data terakhir. Kasus yang umum menggunakan stack adalah pembuatan proses undo/redo pada program.
h. Deque dapat diterapkan untuk stack maupun queue, karena deque dapat memproses data dari depan maupun dari belakang.
i. PriorityQueue merupakan struktur perluasan dari queue. Struktur ini umum digunakan untuk kasus-kasus yang antriannya memiliki kriteria khusus untuk

diakses lebih dahulu. Kasus paling umum menggunakan struktur ini adalah antrian pasien ke dokter. Apabila ada pasien darurat, maka pasti akan didahulukan.

6 DAFTAR PUSTAKA

[1] https://docs.oracle.com.

[2] http://www.Javatpoint.com.

[3] http://beginnersbook.com.

[4] http://www.tutorialspoint.com.