Modul 2 Implementasi Struktur Data
Generic, List dan Array List

2.1. Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:
1. Mengenal konsep Collections pada Java
2. Mengenal konsep ArrayList dan implementasinya pada Java
3. Mengenal konsep Generic dan implementasinya pada Java

2.2. Alat dan Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

2.3. Dasar Teori

Java *Application Programming Interface* (API) menyediakan berbagai tipe struktur data (disebut sebagai collections) yang digunakan untuk menyimpan kumpulan objek yang saling terhubung pada memori. Struktur data ini disimpan pada kelas, dan menyediakan method yang efisien untuk mengatur, menyimpan dan mengambil kembali data tanpa kita perlu mengetahui bagaimana cara data tersebut disimpan. Hal ini akan mengurangi waktu pembangunan program (dan kelak, aplikasi).

Collections mendayagunakan tipe data umum yang dibentuk oleh suatu kelas generic. Selain itu, pada dasarnya collections disusun berdasarkan hirarki interface pada package Java. Pada modul ini akan dipelajari pembentukan kelas generic dan penggunaannya pada dua tipe collections, List dan ArrayList.

2.3.1. Collections

Collection adalah suatu struktur data – bisa dikatakan suatu objek- yang dapat menyimpan reference ke objek lainnya. Collections-framework interface mendeklarasikan operasi yang dapat dilakukan secara general pada berbagai tipe Collections. Tabel 1 merupakan daftar hirarki dari interface Collections pada Java.

Tabel 1 Hirarki Collections

Interface	Keterangan
Collection	Root interface pada hirarki collections darimana interface Set, Queue dan List diturunkan
Set	Collection yang tidak boleh memiliki nilai duplikat
List	Collection tersusun yang dapat memiliki elemen duplikat
Map	Collection yang menghubungkan antara key dan value, tidak boleh memiliki key duplikat. Map tidak diturunkan dari Collection
Queue	Tipe collection ini biasanya berdasarkan susunan first-in, first-out yang dimodelkan dengan jalur antrian

Kelas dan interface pada framework Collections merupakan member dari java.util package. Framework ini menyimpan dan memanipulasi tipe data generic. Generic memungkinkan programmer untuk menspesifikasikan tipe data yang akan disimpan pada Collections. Ketika tipe data tertentu telah dispesifikasikan untuk digunakan pada generic collection, seluruh reference yang berhubungan dengan collection tersebut akan memiliki tipe data tersebut.

Interface Collection memiliki bulk operation (operasi yang terdapat pada seluruh member) untuk beberapa operasi seperti penambahan, penghapusan atau membandingkan objek (atau elemen) dalam suatu collection. Interface ini juga menyediakan method yang mengembalikan objek Iterator, yang mengizinkan program untuk menelusuri collection dan menghapus elemen dari collection tersebut dalam iterasinya. Method lain yang dimiliki oleh interface ini diantaranya dapat menentukan besar dari collection dan apakah suatu collection kosong atau tidak.

Pada modul ini akan dibahas dua tipe kelas Collections yaitu ArrayList dan List.

A. Array List

Sebelum ini, kita menggunakan array untuk menyimpan sekumpulan objek. Namun, jika pada saat program dijalankan ditambahkan elemen diluar besar array yang telah ditetapkan, elemen tersebut tidak dapat ditambahkan ke dalam array. Hal ini disebabkan karena array tidak dapat mengubah ukurannya saat *execution time*. Untuk mengatasi hal ini, Java memiliki kelas collection ArrayList<E> yang dapat mengubah ukurannya secara dinamis untuk mengakomodasi penambahan elemen. “E” (secara konvensi, dipilih huruf “E”) merupakan suatu *placeholder*. Ketika kita mendeklarasikan ArrayList baru, gantilah huruf “E” dengan tipe data yang sesuai. Sebagai contoh, Gambar … memperlihatkan deklarasi ArrayList dengan tipe data String (pada deklarasi ini, huruf “E” diganti dengan String).

```
ArrayList<String> items = new ArrayList<String>();
```

Gambar 1 Deklarasi ArrayList

Tabel 2 menjabarkan beberapa method yang dimiliki oleh kelas ArrayList.

Tabel 2 Method pada ArrayList

Method	Deskripsi
add	Menambah elemen pada akhir atau pada spesifik indeks di ArrayList
clear	Menghapus seluruh elemen dari ArrayList
contains	Mengembalikan nilai *true* jika ArrayList mengandung nilai elemen tertentu, jika tidak akan mengembalikan *false*
get	Mengembalikan elemen pada index tertentu
indexOf	Mengembalikan indeks dari kemunculan pertama dari elemen tertentu di ArrayList
remove	Menghapus nilai tertentu pada kemunculan pertama atau pada indeks tertentu
size	Mengembalikan jumlah elemen yang tersimpan pada ArrayList
trimToSize	Memotong kapasitas ArrayList pada jumlah elemen yang ada saat ini.

Gambar 2 menjelaskan contoh program untuk menyimpan list bermacam bentuk menggunakan ArrayList. Seperti pada array, index ArrayList dimulai dari 0.

```java
package com.company;

import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        ArrayList<String> items = new ArrayList<String>();
        /* or, you can just declare it as this:
          ArrayList<String> items = new ArrayList<>();
        */
        items.add("circle");
        items.add("diamond");
        items.add(1,"star");//add item at index 1 removing an item

        display(items);//before removing an item

        items.add("star");//add another star
        display(items);//displaying list with two stars

        items.remove("star");//remove first occurrence of an item
        display(items);

        System.out.println("Besar ArrayList items: " + items.size());

        System.out.printf("\"rectangle\" is %s in the items list",
                items.contains("rectangle")? " ": "not");

    }

    private static void display(ArrayList<String> list){
        for (String item :list) {
            System.out.print(item+" ");
        }
        System.out.println();
    }
}
```

Gambar 2 Contoh Penggunaan ArrayList

B. List

List (kadang disebut sebagai sequence) adalah kumpulan elemen yang berurutan dan dapat memiliki elemen duplikat. Seperti pada array, indeks pada List juga dimulai dari nol. Interface List diimplementasikan oleh beberapa kelas, diantaranya ArrayList dan LinkedList. Auto-boxing akan terjadi ketika kita menambahkan nilai dengan tipe data primitive pada kelas-kelas tersebut, karena kelas pada List hanya menyimpan reference pada objek.

Kelas ArrayList merupakan implementasi List dari array yang dapat diubah besarnya. Memasukkan elemen diantara elemen yang sudah ada pada ArrayList merupakan operasi yang tidak efisien – seluruh elemen yang terletak sesudah elemen yang baru disisipkan ini harus digeser, yang dapat menjadikan operasi ini operasi yang mahal pada elemen yang berjumlah besar. LinkedList dapat digunakan untuk penyisipan (atau pengahapusan) elemen secara efisien di tengah collection. Namun, LinkedList akan menjadi sangat tidak efisien dibandingkan dengan ArrayList jika harus dilakukan pengaksesan pada data tertentu dalam collection.

Berikut contoh program yang menggunakan ArrayList untuk menyimpan dua array dari color dan menggunakan Iterator untuk menelusuri ArrayList dan menghapus elemen warna pada ArrayList pertama sesuai dengan warna yang terdapat pada ArrayList kedua.

```
package com.company;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;

public class CollectionsDemo {
    public static void main(String[] args) {
        String[] colors = {"Blue", "Cyan", "Navy", "Deep Blue", "Blue Sky"};
        List<String> colorList = new ArrayList<String>();//ArrayList 1

        for(String col: colors)
            colorList.add(col);//add elemen dengan method add yang ada pada List

        String[] removeColors = {"Cyan", "Deep Blue"};
        List <String> removeList = new ArrayList<>>();//ArrayList 2
        for(String col: removeColors)
            removeList.add(col); //add elemen dengan method add yang ada pada List

        System.out.println("Isi Array List sebelum dihapus:");
        for (int i=0; i<colorList.size(); i++) {
            System.out.printf("%s ", colorList.get(i));
        }

        //remove colors from colorList
        remove(colorList,removeList);

        System.out.println("\nIsi Array List sesudah dihapus:");
        for (int i=0; i<colorList.size(); i++) {
            System.out.printf("%s ", colorList.get(i));
        }

    }

    private static void remove(Collection<String> collection, Collection<String>
removedCollection) {
        //get iterator
        Iterator<String> iterator = collection.iterator();
        //loop while collection has item
        while (iterator.hasNext()){
            if(removedCollection.contains(iterator.next())){
                iterator.remove();//remove the item
            }
        }
    }
}
```

2.3.2. Generic Class

Suatu struktur data dapat diimplementasikan tanpa memperhatikan tipe data yang digunakan pada struktur data tersebut. Untuk mencapai hal tersebut, kelas generic digunakan untuk membuat suatu

kelas yang tidak tergantung pada tipe data tertentu. Kemudian, objek dengan tipe data tertentu dapat dibuat dari kelas generic ini.

Hanya tipe data reference yang dapat digunakan untuk mendeklarasikan variable dan membuat objek dari kelas generic. Tipe data primitive seperti int, double, float, boolean dan lainnya tidak dapat digunakan. Namun, Java memiliki mekanisme yang disebut dengan boxing yang dapat mengizinkan tipe data primitive "dibungkus" (wrapped) sebagai objek agar dapat digunakan pada kelas generic. Pembungkusan ini dilakukan dengan kelas type-wrapper dari tipe data primitive, yaitu Boolean, Byte, Character, Double, Float, Integer, Long and Short. Sebagai contoh, Gambar 5 mendeklarasikan suatu ArrayList dengan type-wrapper Integer.

```
ArrayList<Integer> data;
```

Gambar 3 Deklarasi ArrayList dengan Type-Wrapper

Java memiliki mekanisme autoboxing dan auto-unboxing yang akan secara otomatis mengkonversi tipe data primitive menjadi objek type-wrapper nya, dan sebaliknya. Sebagai contoh, array dengan nama "array" memiliki tipe data Integer (type-wrapper dari tipe data integer). Ketika pada array indeks ke-0 dimasukkan nilai integer 10, Java akan otomatis mengkonversi nilai tersebut menjadi Integer. Proses ini disebut dengan auto-boxing. Sementara, ketika nilai dari array[0] disimpan ke dalam variabel value yang memiliki tipe data integer, maka terjadi proses mengubah nilai Integer menjadi integer (Gambar 6).

```
Integer[] array = new Integer[5];//membuat array dengan type-wrapper Integer
array[0] = 10;//auto-boxing integer menjadi Integer
int value = array[0];//auto-unboxing Integer menjadi integer
```

Gambar 4 Proses Auto-boxing dan Auto-unboxing

2.3.3. Studi Kasus

Pada praktikum kali ini, kita akan membuat suatu kelas generic dengan menggunakan ArrayList. Pada kelas generic, dibuat method tambah, remove, display dan edit. Kelas ini kemudian diimplementasikan dengan menggunakan dua tipe data reference Mahasiswa dan Pegawai.

```
package com.company;

import java.util.ArrayList;
import java.util.Collections;



public class GenArrayList<E extends Comparable<? super E>> {
    private final ArrayList<E> list;

    public GenArrayList(int capacity){
        int initCapacity = capacity > 0? capacity:0;
        list = new ArrayList<>(initCapacity);
    }

    public void addData(E values){
        list.add(values);
    }
```

```
    public void display(){
        for (int i = 0; i<list.size();i++) {
            System.out.printf(list.get(i) + " ");
        }
        System.out.println();
    }


    public void displaySort(){
        Collections.sort(list);
        for (int i = 0; i<list.size();i++) {
            System.out.printf(list.get(i) + " ");
        }
        System.out.println();
    }

    public void removeData(E entry){
        list.remove(entry);
    }

    public void setData(E object1, E object2){
        int index = list.indexOf(object1);
        list.add(index, object2);
    }
}
```

Kelas generic ini ditandai dengan notasi diamond <E>. Saat implementasi, notasi "E" ini dapat diganti dengan tipe data apapun, namun tipe data primitive harus diwakili oleh type-wrappernya. Notasi `<E extends Comparable<? super E>>` digunakan karena pada kelas ini dipanggil method sort dari Collection untuk tipe data objek (E). Method add, remove dan set pada kelas ini dibuat hanya untuk tipe data objek. Ketika akan menghapus data, misalnya, yang harus dimasukkan sebagai parameter adalah objek yang ingin dihapus. Konstruktor dari kelas ini menginisiasi besar awal bagi ArrayList.

Pada praktikum kali ini, digunakan dua kelas POJO (Plain Object Java Object) sebagai contoh untuk implementasi kelas generic kelak. Kelas Mahasiswa memiliki atribut nim, nama dan kelas, dengan method toString() dan compareTo() diimplementasikan pada kelas ini.

```
package com.company;

public class Mahasiswa implements Comparable<Mahasiswa> {
    String nim;
    String nama;
    String kelas;

    public Mahasiswa(String nim, String nama, String kelas) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
    }

    public String getNim() {
        return nim;
    }

    public String getNama() {
        return nama;
```

```
    }


    @Override
    public String toString() {
        return nim + " " + nama + " " + kelas;
    }

    @Override
    public int compareTo(Mahasiswa o) {
        return this.getNim().compareTo(o.getNim());
    }

}
```

Kelas Pegawai merupakan kelas Java POJO dengan method yang sama seperti kelas Mahasiswa, tetapi atribut yang dimilikinya adalah nip, nama dan alamat.

```
package com.company;

public class Pegawai implements Comparable<Pegawai> {
    private String nip;
    private String nama;
    private String alamat;

    public Pegawai(String nip, String nama, String alamat) {
        this.nip = nip;
        this.nama = nama;
        this.alamat = alamat;
    }

    public String getNama() {
        return nama;
    }

    @Override
    public String toString() {
        return "Pegawai{" +
                "nip='" + nip + '\'' +
                ", nama='" + nama + '\'' +
                ", alamat='" + alamat + '\'' +
                '}';
    }

    @Override
    public int compareTo(Pegawai o) {
        return this.getNama().compareTo(o.getNama());
    }
}
```

Implementasi dari kelas generic oleh dua kelas POJO ini dilakukan pada kelas Demo. Tipe data bagi objek dari kelas generic mahasiswaList adalah Mahasiswa dan object pegawaiList adalah Pegawai. Kedua tipe data ini dapat digunakan karena kelas generic tidak menspesifikasikan tipe data di awal pembentukannya.

```java
package com.company;

import java.util.Scanner;

public class GenericDemo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        GenArrayList<Mahasiswa> mahasiswaList = new GenArrayList<>(3);
        GenArrayList<Pegawai> pegawaiList = new GenArrayList<>(3);


        for(int i=0; i<3; i++){
            String nip = input.next();
            String nama = input.next();
            String alamat = input.next();

            pegawaiList.addData(new Pegawai(nip,nama,alamat));
        }

        pegawaiList.displaySort();

        for(int i=0; i<3; i++){
            String nim = input.next();
            String nama = input.next();
            String kelas = input.next();
            Mahasiswa mhs = new Mahasiswa(nim, nama, kelas);

            mahasiswaList.addData(mhs);
        }


        mahasiswaList.addData(new Mahasiswa("0", "nina", "22"));
        Mahasiswa mhDelete = (new Mahasiswa("0", "nina", "22"));

        mahasiswaList.displaySort();

        mahasiswaList.removeData(mhDelete);
        mahasiswaList.display();


    }
}
```

Reference

Deitel, P and Deitel H., *Java How to Program: Early Objects*, 11th Ed, Pearson. (2017)