

JAVA COLLECTION DAN GENERIC

Tujuan Pembelajaran

- a. Mengenalkan penggunaan iterator untuk menampilkan elemen atau data collection.
- b. Mengenalkan penggunaan java collection.
- c. Mengenalkan penggunaan generic class.

A. Pengenalan Java Collection

Java Collection API menyediakan pengembang Java dengan satu set kelas dan interface yang membuatnya lebih mudah untuk menangani koleksi data/sekumpulan data. Dalam implementasinya collection sedikit mirip seperti array, kecuali ukuran mereka dapat berubah secara dinamis, dan mereka memiliki lebih maju dari array.

Jika array adalah tempat sekumpulan **elemen**. Maka **Collection** adalah tempat sekumpulan **objek**

Definisi :

Collection adalah sekumpulan interface dan class yang sangat berguna dalam pengolahan variable / objek di Java, bisa dikatakan juga suatu wadah/container yang menampung sekumpulan objek.

Suatu objek collection Java dapat menyimpan beberapa elemen objek dalam suatu kesatuan group.

B. Keunggulan

Dengan pengelompokan dalam group tersebut kita mendapatkan keunggulan berikut :

1. Menyimpan elemen dengan urutan tertentu,
2. Mengambil kembali elemen tersebut dengan index atau dengan berbagai metode yang spesifik,
3. Memanipulasi data terutama agregasi dari elemen-elemen yang ada seperti penjumlahan, dan berbagai transformasi bermanfaat lainnya.

C. Jenis-jenis Interface Java Collection

Interface merupakan tipe data abstract yang dapat mewakili class yang mengimplementasikannya. Jenis-jenis interface yang ada pada Java Collections adalah sebagai berikut :

1. Collection
 - a. Set
 - SortedSet
 - b. List
 - ArraList
 - LinkedList
2. Map
 - HashMap
3. Iterator
 - ListIterator

Berikut adalah matrix perbedaan dari tiap interface secara.

No	Interface	Duplicat	Terurut	Key	Keterangan
1	Collection	Ya	Tidak	Tidak	Sangat generic
2	Set	Tidak	Tidak	Tidak	
3	SortedSet	Tidak	Ya	Tidak	Menggunakan Natural order
4	List	Ya	Sequensial	Index	
5	Queue	Ya	FIFO	Tidak	
6	Map	Tidak		Ya	
7	SortedMap	Tidak	Ya	Ya	

ITERATOR

Iterator adalah interface yang dapat digunakan untuk mengontrol objek yang mengimplementasi interface ini (untuk selanjutnya objek ini disebut objek iterator). Dengan objek iterator, kita dapat menelusuri semua objek yang ada dalam collection. Koleksi dengan jenis yang berbeda memiliki jenis iterator yang berbeda pula, akan tetapi semua iterator digunakan dengan cara yang sama. Algoritma yang menggunakan iterator untuk menelusuri koleksi bisa dibuat generik, karena teknik yang sama bisa digunakan untuk beragam jenis koleksi.

Iterator didefinisikan oleh interface yang bernama Iterator. Interface ini hanya memiliki 3 method diantaranya :

- boolean hasNext();

untuk mengembalikan nilai boolean yang memberi tahu apakah ada item/objek berikutnya bisa diproses atau tidak.

- E next();

mengembalikan item berikutnya, dan memindahkan iterator ke item berikutnya (untuk mendapatkan objek berikutnya). Nilai keluarannya bertipe Object. Ingat bahwa kita tidak bisa melihat suatu item tanpa memindahkan iterator ke item berikutnya. Jika metode ini dipanggil apabila tidak ada item lagi yang tersisa, ia akan melempar NoSuchElementException

- Void remove();

Adapun cirri-ciri method ini :

- dipanggil setelah menjalankan method next()
- menghapus object yang sudah dipanggil dengan next() terakhir
- setelah memanggil next(), tidak bisa menjalankan remove() dua kali, pemanggilan kedua menyebabkan munculnya exception
- Akan melempar exepsi UnsupportedOperationException jika koleksi tidak bisa menghapus item.

Penggunaan interface Iterator dimulai dengan deklarasi Iterator

```
Iterator it = v.iterartor();
```

Setelah itu, pengolahan data dapat dilakukan menggunakan perulangan dan memanfaatkan metode hasNext() dan next().

```
while(v.hasNext()){  
    Object ob =v.next();  
    System.out.println(v);  
}
```

```
for(Iterator i = v.iterator();i.hasNext();){  
    String name = (String) i.next();  
    System.out.println(name);  
}
```

SET

Set Merupakan struktur Data yang digunakan untuk menampung data yang datanya harus unik, makanya datanya hanya ditampung sekali dan juga terurut.

Set Merupakan interface, sehinga dibutuhkan implementasi sebuah kelas untuk menggunakan Set. Dalam java collection implementasi defaultnya adalah HashSet, dimana HashSet melakukan pengecekan duplikasi berdasarkan metode equals() dan hashCode().

Ciri-ciri Set :

- Set extends collection

Set menuruni semua method milik Collection

- Set tidak memperbolehkan adanya duplikasi object di koleksinya

Jika ingin menambah elemen dua kali dengan “add()” maka “add()” yang kedua akan mengembalikan nilai false

- Untuk mengakses object-object di Set, digunakan Iterator
- Representasi Set adalah seperti group, misal group musik, group mata kuliah
- Object di dalam Set tidak urut

Secara default hashCode() setiap objek itu unik dan defaultnya metod equals() membandingkan objek pada memori, jika lokas memorinya sama, maka dianggap sama. Maka dari itu Set sangat berguna sekali apabila kita ingin menampung data yang banyak dan berbeda.

```

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;


public class Main {

public static void main(String[]args){

Set set = new HashSet();


for (int i = 1; i < 5; i ++){

    set.add(i+"-"+KSC laboratory");// menambahkan object ke koleksi

}


System.out.println("ukuran set : "+set.size());

for(Iterator iterator = set.iterator(); iterator.hasNext();){

String ii = (String) iterator.next();

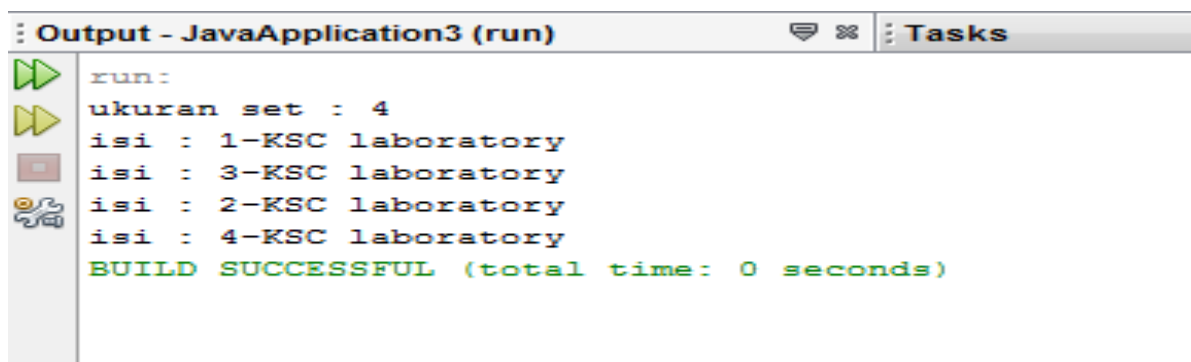
System.out.println("isi : "+ii);

}}

}

```

Dan Hasil Outputnya adalah :



```

run:
ukuran set : 4
isi : 1-KSC laboratory
isi : 3-KSC laboratory
isi : 2-KSC laboratory
isi : 4-KSC laboratory
BUILD SUCCESSFUL (total time: 0 seconds)

```

Pada contoh diatas kita bisa melihat perbedaan yang nyata isi dari data yang disimpan pada Set, namun coba kita modifikasi modifikasi kode program diatas menjadi seperti ini :

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class Main {
    public static void main(String[]args){
        Set set = new HashSet();

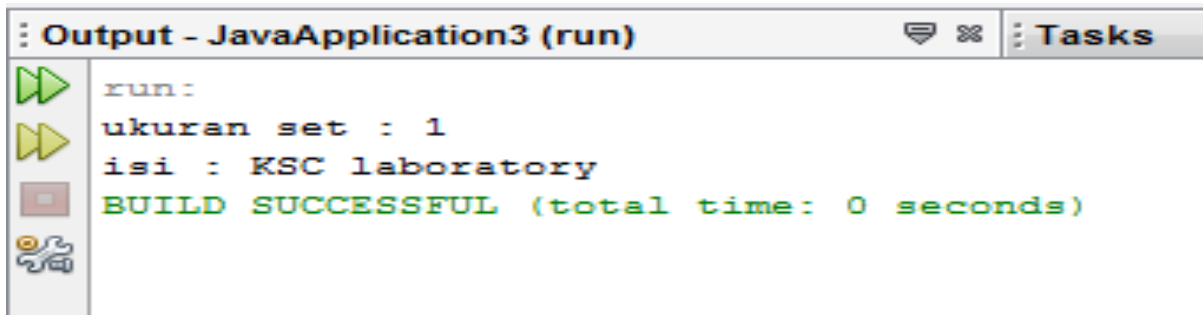
        for (int i = 1; i < 5; i ++) {
            set.add("KSC laboratory");        // menambahkan object ke koleksi
        }

        System.out.println("ukuran set : "+set.size());

        for(Iterator iterator = set.iterator(); iterator.hasNext();){
            String ii = (String) iterator.next();

            System.out.println("isi : "+ii);
        }
    }
}
```

Output yang dihasilkan yaitu :



Maka output yang dihasilkan hanya sebuah String,. padahal tadi ditentukan looping sebanyak lima kali. Hal ini dikarenakan data yang dimasukkan sama, sedang isi Set harus berfifat unik. maka itu hanya di ambil satu kali saja oleh set.

Selain itu kita juga dapat membuat implementasi hashCode sendiri, Misal untuk Mahasiswa, mahasiswa disebut sama jika memiliki nim yang sama, mahasiswa tidak dianggap sama jika Nama yang sama tetapi nimnya berbeda, sehingga kita dapat membuat kelas mahasiswa seperti berikut :

Program di kelas Mahasiswa :

```
package Perkuliahan;

import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class Mahasiswa {
    private int nim;
    private String nama;

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public int getNim() {
        return nim;
    }
}
```

```

    public void setNim(int nim) {

        this.nim = nim;

    }

    @Override

    public boolean equals(Object obj){

        Mahasiswa m= (Mahasiswa) obj;

        return m.nim == nim;

    }

    public int hashCode(){

        return nim;

    }

}

```

LIST

List adalah salah satu interface yang terdapat pada Java collection. Interface ini memiliki beberapa *implementation class* seperti LinkedList, Vector, ArrayList, AbstractList, dan lain-lain.

Ciri dari List adalah :

- objek-objeknya memiliki **urutan tertentu** untuk mengaksesnya (*ordered*) seperti dengan penggunaan nomor **index** atau dengan melakukan pencarian berdasarkan nilai tertentu dari objek. Urutan ini bisa berbeda dari tiap class yang mengimplementasikan interface List ini.
- Nomor index yang digunakan dimulai dari angka **0** (*zero based index*).
- Elemen-elemennya bisa memiliki objek yang sama (**duplikat**).

Program di Main :

```
package perkuliahan;

import Perkuliahan.Mahasiswa;
import java.util.HashSet;
import java.util.Set;

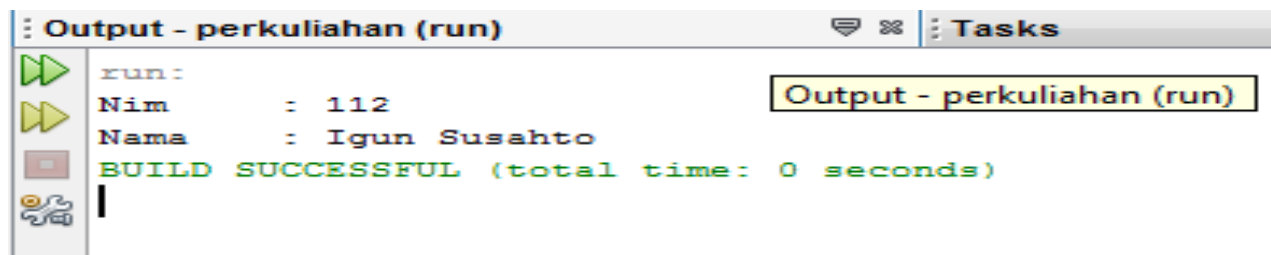
public class Main {
    public static void main(String[] args) {
        Set<Mahasiswa> set= new HashSet<Mahasiswa>();

        Mahasiswa a = new Mahasiswa();
        a.setNama("Igun Susahto");
        a.setNim(112);
        set.add(a);

        Mahasiswa b = new Mahasiswa();
        b.setNama("agoes Soedirmoan");
        b.setNim(112);
```

```
for(Mahasiswa m : set){  
    System.out.println("Nim    : "+m.getNim());  
    System.out.println("Nama   : "+m.getNama());  
}  
}  
}
```

Maka Output yang dihasilkan :



```
run:  
Nim      : 112  
Nama     : Igun Susahto  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Dengan demikian hashCode nya akan dicetak berdasarkan Nim.

SORTED SET

SortedSet memiliki fitur yang sama dengan Set, namun object yang tersimpan diurutkan berdasarkan datanya. Efek penggunaan SortedSet, proses penambahan data jadi lebih lambat daripada Set biasa

```
import java.util.Iterator;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;

public class Main {

    static private void settingSet(SortedSet<Integer> set) {

        set.removeAll(set);    // kosongkan isi koleksi

        for (int i = 10000; i > 0; i /= 2) {

            set.add(i); // menambah object ke koleksi

        }
        //secara descending
    }

    static private void printSet(SortedSet<Integer> set) {

        Iterator itr = set.iterator(); // mendapatkan object iterator dari
        koleksi

        while (itr.hasNext())

            System.out.print(itr.next() + " "); // menampilkan isi
        koleksi

        System.out.println();

    }

}
```

```

static public void main(String[] args) {

    SortedSet<Integer> set = new TreeSet<Integer>();    //
    membuat object TreeSet

    // kemudian dihandle oleh SortedSet

    settingSet(set);          // setting isi koleksi

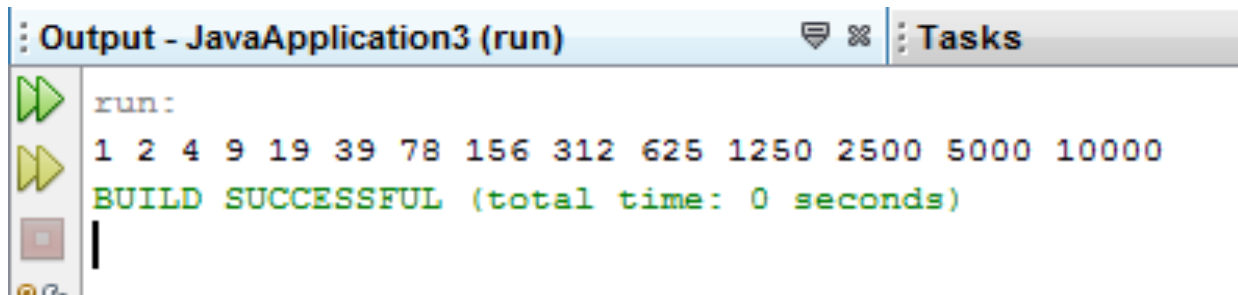
    printSet(set);            // tampilkan koleksi

}

}

```

Outputnya :



```

run:
1 2 4 9 19 39 78 156 312 625 1250 2500 5000 10000
BUILD SUCCESSFUL (total time: 0 seconds)

```

Jadi meskipun penambahan object integer ke koleksi dimulai dari nilai besar ke nilai kecil, object yang disimpan otomatis diorganisasi urut dari kecil ke besar

LIST

List adalah salah satu interface yang terdapat pada Java collection. Interface ini memiliki beberapa *implementation class* seperti LinkedList, Vector, ArrayList, AbstractList, dan lain-lain.

Ciri dari List adalah :

- List extends Collection (Member dari interface List di turunkan dari Collection).
- objek-objeknya memiliki **urutan tertentu** untuk mengaksesnya (*ordered*) seperti dengan penggunaan nomor **index** atau dengan melakukan pencarian berdasarkan nilai tertentu

dari objek. Urutan ini bisa berbeda dari tiap class yang mengimplementasikan interface List ini. Kesimpulannya Items/Objects di List dapat diakses menggunakan posisinya / indeksnya

- Nomor index yang digunakan dimulai dari angka **0** (*zero based index*).
- Elemen-elemennya bisa memiliki objek yang sama (**duplikat**).
- List cocok digunakan untuk mengganti Array, dengan ukuran yang dinamis

Ada dua cara paling umum untuk membuat list : sebagai array dinamis dan sebagai list berantai :

1. ArrayList

ArrayList adalah urutan objek yang disimpan dalam bentuk array yang ukurannya bisa membesar jika item baru ditambahkan

2. LinkedList

LinkedList adalah urutan objek yang disimpan dalam simpul yang terhubung dengan pointer seperti rantai.

ArrayList

Interface List menambah beberapa metode untuk mengakses item pada list tergantung dari urutannya dalam list. metode-metode yang tersedia terutama dalam ArrayList adalah :

- **public E get(int index)**

mengembalikan Object di posisi indeks dalam list, di mana indeks dimulai dari 0, 1, 2, hingga list.size() - 1. Parameter indeks harus ada dalam rentang ini, jika tidak maka pengecualian IndexOutOfBoundsException akan dilemparkan.

- **public E set(int index, E x)**

menyimpan obj di dalam list pada posisi indeks, dan mengganti isi objek yang sudah ada di posisi tersebut. Metode ini tidak mengubah jumlah elemen di dalam list atau memindahkan elemen lain.

- **public void add(int index, E x)**

menyisipkan objek obj di dalam list pada posisi indeks. Jumlah item di dalam list akan bertambah satu, dan item setelah posisi indeks akan digeser ke belakang. Nilai indeks harus berada pada rentang 0 hingga list.size().

- **public E remove(int index)**

menghapus objek pada posisi indeks. Item setelah posisi ini akan digeser maju untuk mengisi kekosongan objek pada posisi tersebut setelah item dihapus.

- **public int indexOf(Object e)**

mencari objek obj di dalam list dan mengembalikan indeks nya pada list, jika ada. Jika objek yang dicari tidak ada maka nilai -1 akan dikembalikan. Jika obj ada lebih dari satu dalam list, hanya indeks pertama yang dikembalikan

Contoh implementasi ArrayList

```
package javaapplication3;

import java.util.ArrayList;
import java.util.List;

public class Main {

    private static void loadData(List<String> list) {

        list.add("nol");

        list.add("satu");

        list.add("dua");

        list.add("tiga");

        list.add("empat");

    }

}
```

```

private static void tampilkanList(List<String> list) {

    for (int i = 0; i < list.size(); i++)

        System.out.print(list.get(i) + " "); // menampilkan isi
koleksi

        System.out.println();

    }

    public static void main(String[] args) {

        List<String> list = new ArrayList<String>();

        loadData(list);

        tampilkanList(list);

        System.out.println("list di index ke 1 = "+list.get(0));

    }

}

```

Output yang dihasilkan :

```

Output - JavaApplication3 (run)
run:
nol satu dua tiga empat
list di index ke 1 = nol
BUILD SUCCESSFUL (total time: 0 seconds)

```

Jadi berdasarkan program diatas kita bisa mengakses elemen yang ada pada List dengan menggunakan indeksinya.

Kekurangan ArrayList yaitu :

Saat kita menghapus data maka data pada index setelah data yang dihapus , akan diubah indexnya ke index sebelumnya. Misalnya kita memiliki 100 data dalam ArrayList, lalu kita menghapus data index ke 20, maka proses yang terjadi adalah :

1. Hapus data ke 21
2. Tempatkan data ke 22 menjadi ke 21

3. Tempatkan data ke 23 menjadi ke 22
4. Dan seterusnya.

Sehingga ArrayList tidak cocok untuk menampung data yang dalam berjalannya sering terjadi proses penghapusan data.

LinkedList

Berbeda dengan Halnya ArrayList data tidak disimpan dalam index. LinkedList merupakan solusi yang dapat menutup kekurangan ArrayList yaitu pada saat penghapusan data hanya melakukan 3 langkah, misal kita menghapus data ke 5 :

1. Hapus data ke 5.
2. Ubah pointer next data ke 4 mengacu ke data 6.
3. Ubah pointer prev data ke 5 menjadi mengacu ke data 4.

Seluruh method yang ada pada ArrayList tersedia juga pada LinkedList, hanya saja kelas LinkedList memiliki beberapa metode tambahan yang tidak ada pada ArrayList. Jika linkedlist adalah objek bertipe LinkedList, maka :

- `linkedlist.getFirst()`

Mengembalikan Object pada posisi pertama di dalam list. List tidak diubah sama sekali.

- `linkedlist.getLast()`

Mengembalikan Object pada posisi terakhir di dalam list. List tidak diubah sama sekali.

- `linkedlist.removeFirst()`

Menghapus Objek pada posisi pertama di dalam list. Object yang dihapus akan dikembalikan sebagai nilai keluaran.

- `linkedlist.removeLast()`

Menghapus Objek pada posisi terakhir di dalam list. Object yang dihapus akan dikembalikan sebagai nilai keluaran.

- `linkedlist.addFirst(obj)`

Menambah obj pada posisi pertama di dalam list.

- `linkedlist.addLast(obj)`

Menambah obj pada posisi terakhir di dalam list. (Sama persis dengan linkedlist.add(obj))

Contoh program Implementasi LinkedList :

```
package javaapplication3;

import java.util.*;

public class Main {

    public static void main(String[] args) {

        List<String> list = new LinkedList();//deklarasi pembuatan List

        for(int i=0; i<=3; i++){

            if(i % 2 == 0){

                list.add("- Laboratorium KSC");//untuk memasukkan data ke
                dalam List jika i = genap

            }else{

                list.add("- KSC laboratory");//untuk memasukkan data ke dalam
                List jika i = ganjil

            }

        }

        System.out.println("Ukuran List --> "+list.size());//untuk melihat
        jumlah data di dalam List

        Collections.sort(list);// untuk men sortir data di dalam list

        for(Iterator<String> iterator = list.iterator();
        iterator.hasNext();){//untuk melihat isi List

            String isi = iterator.next();

            System.out.println(isi);

        }

    }

}
```

Kekurangan LinkedList :

Saat proses pencarian dikarenakan pada LinkedList, data tidak memiliki index, maka saat terjadi proses pencarian, maka dilakukan secara sequensial, sehingga dapat memperlambat proses pencarian and memakan memory cukup besar.

MAP

Sering juga disebut dictionary karena objek disimpan berpasangan dengan kunci(key). Artinya, setiap objek yang disimpan dalam maps mempunyai kunci yang berbeda. Sebagai contoh, kita memiliki suatu katalog buku, kode buku adalah kunci dan object buku adalah datanya.

Map sangat cocok digunakan pada data cukup kompleks. Dengan demikian, programmer tidak harus menghafal letak index seperti pada array dan collection class sequence lainnya

Ciri-ciri Map :

- Kunci di Map tidak dapat diduplikasi
- Tiap kunci hanya menunjuk ke satu object data
- Penggunaan Map biasanya dipadukan dengan class HashMap

Method yang ada di Map:

Metode	Keterangan
void clear()	Menghapus semua elemen dalam HashMap sehingga ukurannya menjadi 0.
boolean isEmpty()	Nilai true dikembalikan jika tidak ada elemendi dalam.
int size()	Mengembalikan jumlah elemen dalam HashMap.
boolean containsKey(Object key)	Nilai True jika key ditemukan dalam HashMap.
boolean containsValue(Object value)	Nilai true jika value ditemukan dalam HashMap.
object get(Object key)	Mengembalikan nilai Object dengan key tertentu.
setkeySet()	Mengambil key dan dimasukkan ke dalam Set. Key tidak

	<p> mungkin ada yang sama sehingga tidak menimbulkan masalah jika diubah ke dalam format Set. </p>
<p> object put(Object key, Object value). </p>	<p>Menambah elemen ke dalam HashMap.</p>
<p>object remove(Object key)</p>	<p>Membuang objek dari HashMap.</p>

Berikut contoh penggunaan dan pengolahan data dengan Map :

```

import java.util.Collection;

import java.util.HashMap;

import java.util.Set;


public class Main {

    public static void main(String[] args) {

        HashMap map = new HashMap();

        map.put("Nama","joko");

        map.put("NIM",new Integer(9523257));

        map.put("Alamat", "kaliurang");

        System.out.println("Ukuran Map : "+map.size());


        map.put("Email", "joko@yahoo.com");

        Object email = map.get("Email");

        System.out.println("Email nya : "+email);


        boolean containKey = map.containsKey("NIM");

        System.out.println("Has key (NIM): "+containKey);
    }
}

```

```

        Object removed = map.remove("NIM");

        System.out.println("Removed : "+removed);

        System.out.println("Size baru : "+map.size());

        Set set = map.keySet();

        System.out.println("Key Set size: "+set.size());

        boolean removedFromSet = set.remove("Alamat");

        System.out.println("Alamat: "+removedFromSet);

        System.out.println("Size: "+map.size());

        Collection values = map.values();

        System.out.println("Collection size: "+values.size());

    }

}

```

Generic

Generic merupakan implementasi tipe data pada koleksi. Tanpa adanya generic, kita dapat memasukkan tipe data yang berbeda-beda dalam sebuah koleksi. Baru ketika data tersebut diambil, maka perlu dilakukan casting.

Misal pada koleksi ArrayList, objek dapat ditambahkan dengan perintah berikut :

```

public boolean add(Object o){
    //statement(s)
}

```

Sedangkan untuk pengambilan data, harus dilakukan casting tipe data sebagai berikut :

```

Mahasiswa e = (Mahasiswa) organisasi.get(0);

```

Artinya, koleksi ArrayList organisasi memiliki elemen dengan tipe objek Mahasiswa. Masalah dapat timbul jika ada beberapa elemen yang bukan bertipe Mahasiswa. Elemen lain mungkin saja ada karena semua Object dapat ditambahkan dengan metode add().

Deklarasi dan Penerapan Generic

Tipe generic pada koleksi dapat diterapkan dengan menambahkan tanda<>.

```

ArrayList<Mahasiswa> organisasi;

```

```
Vector<integer> kodePos = new Vector<Integer>;
```

Jika dipaksakan menambahkan elemen dengan tipe yang berbeda maka akan keluar error misal :

```
kodePos.add("belum ada"); //akan terjadi error
```

Dengan adanya generic, program lebih handal karena dapat mencegah kesalahan-kesalahan karena kelalaian programmer.

Latihan

```
1 public class Barang {
2
3     private String nama_barang;
4     private int jumlah;
5
6     public Barang(String nama_barang, int jumlah) {
7         this.nama_barang = nama_barang;
8         this.jumlah = jumlah;
9     }
10
11     public int getJumlah() {
12         return jumlah;
13     }
14
15     public void setJumlah(int jumlah) {
16         this.jumlah = jumlah;
17     }
18
19     public String getNama_barang() {
20         return nama_barang;
21     }
22
23     public void setNama_barang(String nama_barang) {
24         this.nama_barang = nama_barang;
25     }
26 }
```

```
1 import java.util.ArrayList;
2
3 public class KeranjangBelanja <t> {
4
5     private ArrayList<t> data;
6     private String nama;
7     private t item;
8
9     public KeranjangBelanja(String nama) {
10         this.nama=nama;
11         this.data=new ArrayList<t>();
12
13     }
14
15     public String getNama() {
16         return nama;
17     }
18 }
```

19	
20	
21	public ArrayList<t> data() {
22	return data;
23	}
24	
25	public t getItem(int idx) {
26	return this.data().get(idx);
27	}
28	
29	
30	
31	public void addItem(t item)
32	{
33	
34	this.data.add(item);
35	}
	}

1	public class Main {
2	
3	
4	public static void main(String[] args) {
5	
6	KeranjangBelanja <Barang> sc = new KeranjangBelanja("fadil");
7	
8	sc.addItem(new Barang("Big Cola", 3));
9	
10	sc.addItem(new Barang("Silver Queen", 7));
11	
12	System.out.println("nama :" + sc.getNama());
13	
14	System.out.println("Pesanan anda ");
15	
16	for(Barang b: sc.data())
17	{
18	System.out.println("nama barang :"+b.getNama_barang());
19	System.out.println("Jumlah :"+b.getJumlah());
20	}
21	
22	}
23	
24	}
25	
26	

Pertanyaan :

1. Collection apa yang digunakan pada kelas KeranjangBelanja ?
2. Pada kelas KeranjangBelanja apa maksud baris ke 34?
3. Pada kelas KeranjangBelanja apa maksud baris ke 26?
4. Pada kelas KeranjangBelanja ubah Collection ArrayList menjadi List dengan menambah package java.util.List, apa yang terjadi? Jelaskan
5. Pada kelas KeranjangBelanja ubah Collection ArrayList menjadi LinkedList apa yang terjadi
6. Pada kelas Main apa maksud dari baris ke 6 ?
7. Pada kelas Main jelaskan maksud baris ke 16 ?
8. Ubahlah bentuk dibawah ini menjadi perulangan biasa untuk menampilkan elemen yang ada.

```
for (Barang b: sc.data())  
{  
    System.out.println("nama barang :"+b.getNama_barang());  
    System.out.println("Jumlah :"+b.getJumlah());  
}
```

9. Pada kelas Main, tambahkan isi sc hingga elemen yang dimilikinya mencapai 5!
10. tampilkan elemen pada List ke0 s.d ke-3!