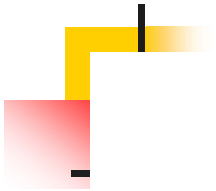
Collection

Oleh:
Mike Yuliana
PENS-ITS

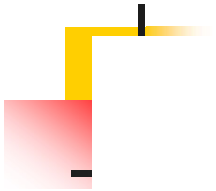
Topik

- Collections Framework: Collections API
- Interface Collections API:
  - Collection
  - List
  - Set
- Map
- Retrieve elements:
  - Iterator
  - ListIterator
  - Enumeration



Collections Framework

- Dikenalkan pada Java 2 SDK.
- Collection sudah ada sejak JDK 1.0
  - Hashtable
  - Vector



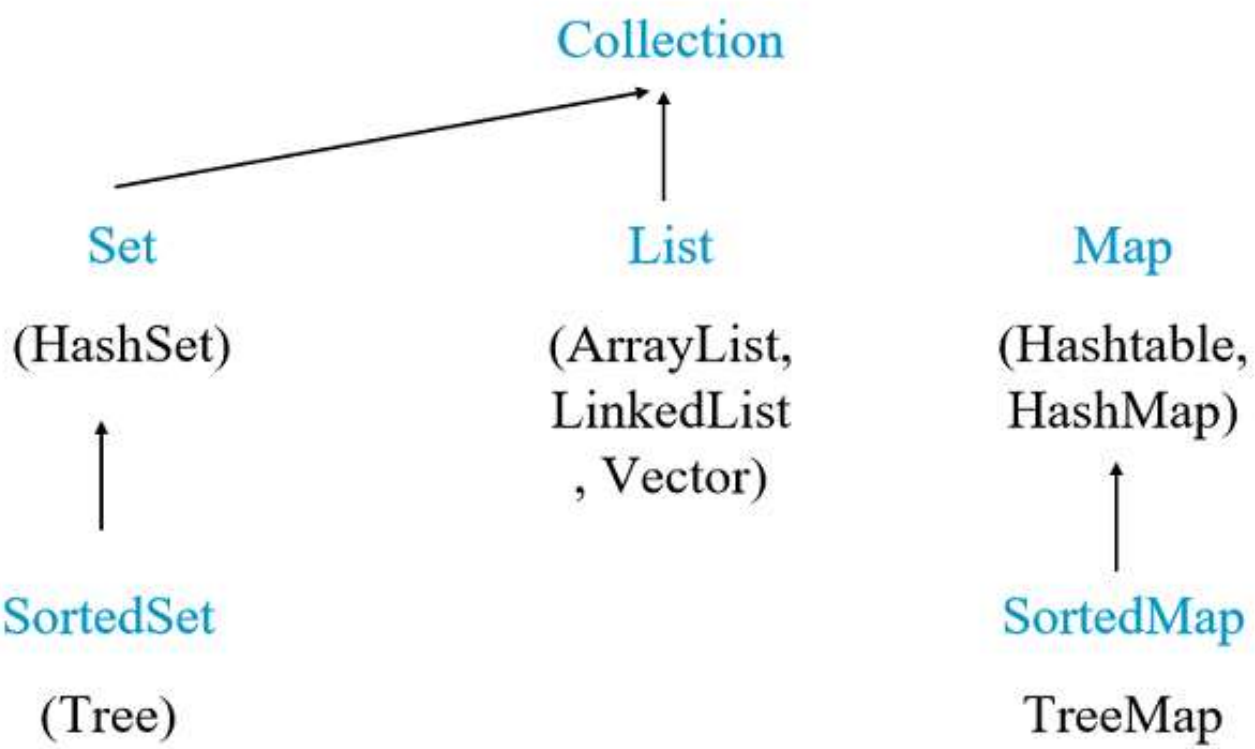
The Java Collections API

- Collection adalah suatu obyek yang bisa digunakan untuk menyimpan sekumpulan obyek
- Obyek yang ada dalam collection ini disebut sebagai elemen.
- Collection menyimpan elemen yang bertipe Object, sehingga berbagai tipe obyek bisa disimpan dalam collection.

The Java Collections API

- Java Collections API terdiri dari interface:
  - **`Collection`** : sekumpulan obyek yang tidak mempunyai posisi yang tetap (no particular order) dan menerima duplikat.
  - **`List`**: sekumpulan obyek yang urut (ordered) dan menerima duplikat.
  - **`Set`**: sekumpulan obyek yang tidak urut (unordered) dan menolak duplikat.
  - **`Map`**: mendukung pencarian berdasarkan key, key ini harus unik. Has no particular order.

Interface Collection dan Hirarki Class



Method Interface Collection

- boolean add(Object element)

  Menambahkan elemen pada collection, bila berhasil akan mengembalikan nilai true.
- boolean remove(Object element)

  Menghapus elemen di collection, bila berhasil akan mengembalikan nilai true.
- int size()

  Mengembalikan jumlah elemen yang terdapat pada collection.

- boolean isEmpty()

  Jika tidak terdapat elemen sama dalam collection maka akan mengembalikan nilai true.
- boolean contains(Object elemen)

  Akan mengembalikan nilai true jika elemen terdapat pada collection.
- boolean containsAll(Collection collection_A)

  Akan mengembalikan nilai true jika semua elemen yang ada pada collection_A ada pada collection.

- boolean addAll(Collection collection)

  Akan mengembalikan nilai true jika semua elemen yang ada pada collectionA berhasil ditambahkan pada collection.
- void clear()

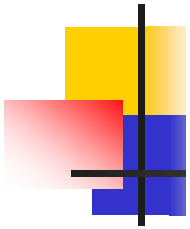
  Mengapus semua elemen collection.
- void removeAll(Collection collection_A)

  Menghapus semua elemen collection yang ada pada collectionA
- void retainAll(Collection collection_A)

  Menghapus semua elemen Collection kecuali elemen yang ada pada Collection_A

Set: HashSet

- Elemen pada Set selalu unik.
- Set menolak duplikat.
- Elemen yang tersimpan tidak urut (unordered) dan unsorted.
- Interface Set merupakan sub interface dari interface Collection.
- Contoh Set: HashSet.



Set: HashSet

```
import java.util.*;

public class SetExample {
    public static void main(String[] args) {
        Set set = new HashSet();
        set.add("one");
        set.add("second");
        set.add("3rd");
        set.add(new Integer(4));
        set.add(new Float(5.0F));
        set.add("second"); // duplicate, not added
        set.add(new Integer(4)); // duplicate, not added
        System.out.println(set);
    }
}
```

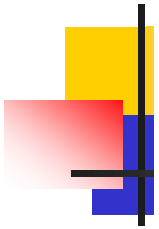
Hasil:

```
[one, second, 5.0, 3rd, 4]
```

SortedSet:TreeSet

- Aturan sama dengan interface Set → menolak duplikat.
- Ingat → SortedSet adalah subinterface Set.
- Beda: elemen tersimpan dalam urutan ascending → sorted.
- Contoh SortedSet: TreeSet.

SortedSet: TreeSet

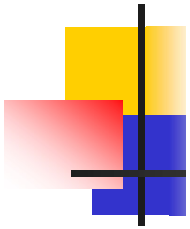


```
import java.util.*;
class SortedSetTest{
    public static void main(String [] arg){
        SortedSet set = new TreeSet();
        set.add("Chess");
        set.add("Whist");
        set.add("Checkers");
        set.add("BlackJack");
        set.add("Chest");
        System.out.println(set);
    }
}
```

Output: [BlackJack, Checkers, Chess, Whist]

Contoh Soal

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

/**
 *
 * @author HP
 */
public class dua {
    public static void main(String[] args) {
        SortedSet set=new TreeSet();
        set.add("chess");
        set.add("whist");
        set.add("Checkers");
        set.add("blackjack");
        set.add("chess");
        System.out.println(set);
    }

}
```



List

- Elemen tersimpan terurut (ordered).
- Urut berdasarkan masukan.
- Menerima duplikat.
- Contoh List:
  - LinkedList : elemen dalam LinkedList masuk dari awal dan dihapus dari akhir.
  - Vector : a growable array of object.
  - ArrayList: mirip vector, bersifat unsyncronized (jika multiple threads mengakses object ArrayList, object ini harus syncronized secara eksternal)

List : ArrayList

```
1   import java.util.*
2
3   public class ListExample {
4     public static void main(String[] args) {
5       List list = new ArrayList();
6       list.add("one");
7       list.add("second");
8       list.add("3rd");
9       list.add(new Integer(4));
10      list.add(new Float(5.0F));
11      list.add("second");          // duplicate, is added
12      list.add(new Integer(4));    // duplicate, is added
13      System.out.println(list);
14    }
15  }
```

```
[one, second, 3rd, 4, 5.0, second, 4]
```

List: Vector

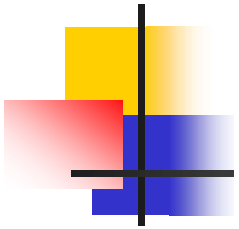
```
import java.util.*;
class VectorTest{
    public static void main(String [] arg){
        Vector v = new Vector();
        v.add("Zak");
        v.add("Gordon");
        v.add(0 ,"Duke");
        v.add("Lara");
        v.add("Zak");
        System.out.println(v);
        String name = (String) v.get(2);
        System.out.println(name);
    }
}
```

Output: [Duke, Zak, Gordon, Lara, Zak]

Gordon

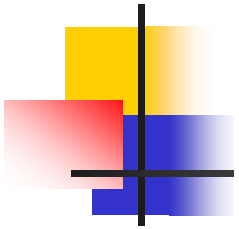
Contoh Soal

```
import java.util.Vector;

/**
 *
 * @author HP
 */
public class empat {
    public static void main(String[] args) {
        Vector v=new Vector();
        v.add("zak");
        v.add("gordon");
        v.add(1,"duke");
        v.add(3,"lara");
        v.add("zak");
        System.out.println(v);
        String name=(String)v.get(2);
        System.out.println(name);

    }
```



Map

- Menyimpan elemen dengan key unik.
- Satu key untuk satu elemen.
- Key disimpan dalam bentuk object.
- Map tidak bisa menyimpan duplicate key.
- Map bisa menyimpan duplicate element.
- Has no particular order.
- Contoh:
  - Hashtable
  - HashMap
    - not syncronized for threads
    - permits null values to be stored



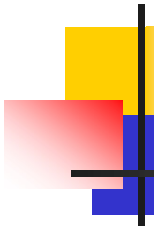
Map: HashMap

```
import java.util.*;
class HashMapTest{
    public static void main(String [] arg){
        HashMap hm = new HashMap();
        hm.put("Game1", "Hearts");
        hm.put(null, "Chess");
        hm.put("game3", "Checkers");
        hm.put("game3", "Whist");
        hm.put("game4", "Chess");
        System.out.println(hm);
    }
}
```

Output: {Game4=Chess, Game3=Whist, Game1=Hearts, null=Chess}

Contoh Soal

```
public class lima {
    public static void main(String[] args) {
        HashMap hm=new HashMap();
        hm.put("game 1", "hearts");
        hm.put(null, "chess");
        hm.put("game 3","checkers");
        hm.put("game 3", "whist");
        hm.put("game 4", "jack");
        hm.put("game 4", "chess");
        hm.put("game 3", "jaki");
        System.out.println(hm);
    }

}
```



SortedMap: TreeMap

- Aturan mirip Map
- Beda: obyek tersimpan secara sorted berdasarkan key.
- No duplicate key.
- Elements may be duplicate.
- Key tidak boleh null value.

SortedMap: TreeMap

```
import java.util.*;
class TreeMapTest{
    public static void main(String [] args){
        SortedMap title = new TreeMap();
        title.put(new Integer(3), "Checkers");
        title.put(new Integer(1), "Euchre");
        title.put(new Integer(4), "Chess");
        title.put(new Integer(2), "Tic Tac Toe");
        System.out.println(title);
    }
}
```

Output: {1=Euchre, 2=Tic Tac Toe, 3=Checkers, 4=Chess}