
JAVA COLLECTION FRAMEWORK : MAP

A. TUJUAN PEMBELAJARAN

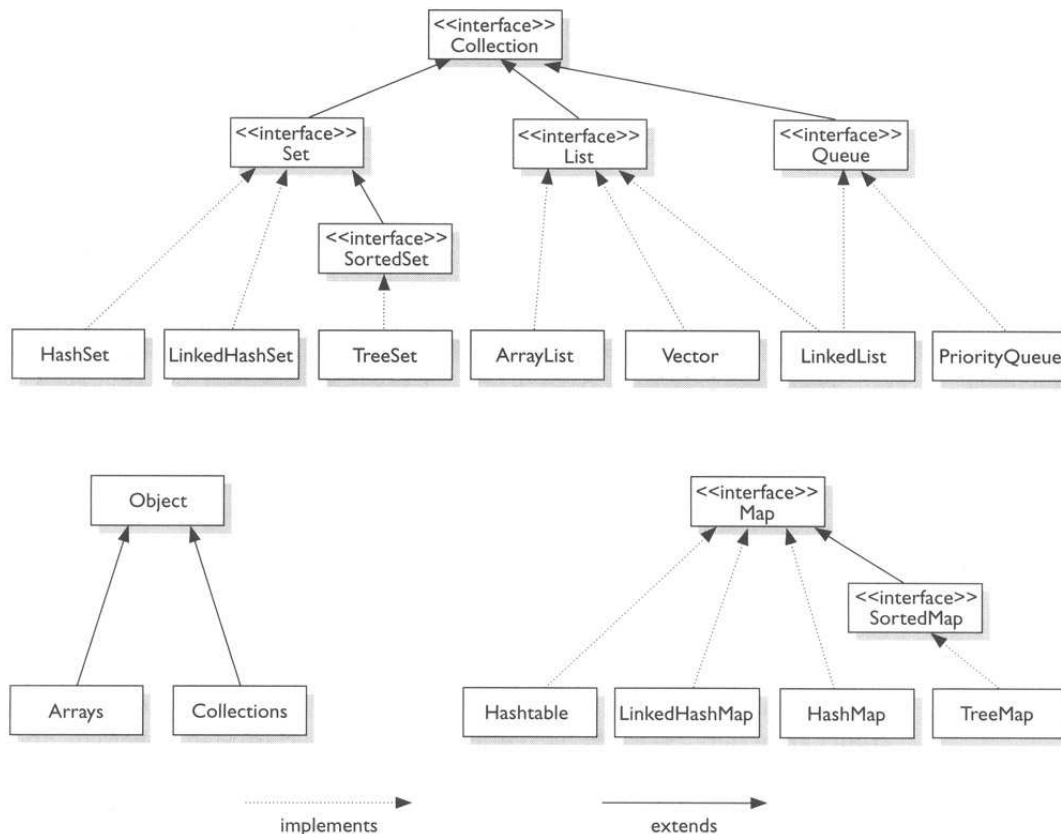
1. Mengetahui mengenai interface Map.
2. Memahami cara penyimpanan objek menggunakan Map.
3. Mengetahui implementasi penggunaan dari class-class pada interface Map.

B. DASAR TEORI

Collection adalah suatu objek yang bisa digunakan untuk menyimpan sekumpulan objek. Objek yang ada dalam Collection disebut elemen. Collection menyimpan elemen yang bertipe Object, sehingga berbagai tipe objek bisa disimpan dalam Collection.

Class-class mengenai Collection tergabung dalam Java Collection Framework. Class-class Collection diletakkan dalam package `java.util` dan mempunyai dua interface utama yaitu `Collection` dan `Map`. Mulai java 1.5 (juga dikenal sebagai J2SE 5), semua class yang termasuk Java Collection Framework adalah class generics. Untuk kompatibilitas dengan versi java sebelumnya, penggunaan generics tidak diharuskan, namun sangat disarankan.

Collection terbagi menjadi 3 kelompok yaitu Set, List dan Map. Berikut ini adalah struktur hierarki interface dan class yang termasuk dalam kelompok collection ini.



Java Collections Framework terbagi menjadi tiga kelompok:

- **Set**

Set mengikuti model himpunan, dimana objek/anggota yang tersimpan dalam Set harus unik. Urutan maupun letak dari anggota tidaklah penting, hanya keberadaan anggota saja yang penting. Class-class yang mengimplementasikan interface Set adalah HashSet. Interface SortedSet merupakan subInterface dari interface Set. Untuk mengurutkan Set, kita dapat menggunakan class yang mengimplementasikan interface SortedSet yaitu class TreeSet.

- **List**

List digunakan untuk menyimpan sekumpulan objek berdasarkan urutan masuk (ordered) dan menerima duplikat. Cara penyimpanannya seperti array, oleh sebab itu memiliki posisi awal dan posisi akhir, menyisipkan objek pada posisi tertentu, mengakses dan menghapus isi list, dimana semua proses ini selalu didasarkan pada urutannya. Class-

class yang mengimplementasikan interface List adalah Vector, Stack, Linked List dan Array List.

Terdapat interface Queue yang cara penyimpanan seperti List, interface ini menyimpan objek menggunakan metode FIFO (First In First Out) yaitu objek yang masuk pertama keluar pertama. Class-class yang mengimplementasikan interface Queue adalah PriorityQueue dan LinkedList. Data yang tersimpan pada objek PriorityQueue akan diurutkan, data tersebut harus mengimplementasikan objek Comparable atau Comparator.

- **Map**

Perbedaan mendasar map dengan collection yang lain, untuk menyimpan objek pada Map, perlu sepasang objek, yaitu key yang bersifat unik dan nilai yang disimpan. Untuk mengakses nilai tersebut maka kita perlu mengetahui key dari nilai tersebut. Map juga dikenal sebagai dictionary/kamus. Pada saat menggunakan kamus, perlu suatu kata yang digunakan untuk pencarian. Class-class yang mengimplementasikan Map adalah Hashtable, HashMap, LinkedHashMap. Untuk mengurutkan Map menggunakan interface SortedMap, class yang mengimplementasikan interface tersebut adalah TreeMap.

C. TUGAS PENDAHULUAN

Buatlah resume 1 halaman mengenai interface Map dan 2 contoh dari interface Map.

D. PERCOBAAN

Percobaan 1 : Penggunaan HashMap, menambahkan data, menghapus data tertentu dan menghapus semua data pada objek HashMap.

```
import java.util.HashMap;

public class RemoveValueFromHashMapExample {

    public static void main(String[] args) {
        //create HashMap object
        HashMap hMap = new HashMap();

        //add key value pairs to HashMap
        hMap.put("1", "One");
        hMap.put("2", "Two");
    }
}
```

```

    hMap.put("3", "Three");
    Object obj = hMap.remove("2");
    System.out.println(obj + " Removed from HashMap");
    hMap.clear();

    System.out.println("Total key value pairs in HashMap are : " +
hMap.size());
    }
}

```

Percobaan 2 : Melakukan iterasi pada value HashMap

```

import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;

public class IterateValuesOfHashMapExample {

    public static void main(String[] args) {
        //create HashMap object
        HashMap hMap = new HashMap();

        //add key value pairs to HashMap
        hMap.put("1", "One");
        hMap.put("2", "Two");
        hMap.put("3", "Three");

        Collection c = hMap.values();
        //obtain an Iterator for Collection
        Iterator itr = c.iterator();

        //iterate through HashMap values iterator
        while(itr.hasNext())
            System.out.println(itr.next());
    }
}

```

Percobaan 3 : Mendapatkan key, melakukan iterasi pada key dan menghapus key tertentu pada objek HashMap

```

public class GetSetViewOfKeysFromHashMapExample {

    public static void main(String[] args) {
        //create HashMap object
        HashMap hMap = new HashMap();

        //add key value pairs to HashMap
        hMap.put("1", "One");
        hMap.put("2", "Two");
        hMap.put("3", "Three");
    }
}

```

```

Set st = hMap.keySet();

System.out.println("Set created from HashMap Keys contains :");

//iterate through the Set of keys
Iterator itr = st.iterator();
while(itr.hasNext())
    System.out.println(itr.next());

//remove 2 from Set
st.remove("2");
}
}

```

Percobaan 4 : Mengecek apakah objek HashMap mempunyai value tertentu.

```

import java.util.HashMap;

public class CheckValueOfHashMapExample {
    public static void main(String[] args) {

        //create HashMap object
        HashMap hMap = new HashMap();

        //add key value pairs to HashMap
        hMap.put("1", "One");
        hMap.put("2", "Two");
        hMap.put("3", "Three");

        boolean blnExists = hMap.containsValue("Two");

        System.out.println("Two exists in HashMap ? : " + blnExists);
    }
}

```

Percobaan 5 : Mengecek apakah objek HashMap berisi key tertentu

```

import java.util.HashMap;

public class CheckKeyOfHashMapExample {

    public static void main(String[] args) {
        //create HashMap object
        HashMap hMap = new HashMap();

        //add key value pairs to HashMap
        hMap.put("1", "One");
        hMap.put("2", "Two");
        hMap.put("3", "Three");

        boolean blnExists = hMap.containsKey("3");
    }
}

```

```
        System.out.println("3 exists in HashMap ? : " + blnExists);
    }
}
```

Percobaan 6 : Menambahkan objek Hash Map ke objek Hashtable dan penggunaan Enumeration.

```
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.HashMap;

public class CreateHashtableFromHashMap {

    public static void main(String[] args) {
        //create HashMap
        HashMap hMap = new HashMap();

        //populate HashMap
        hMap.put("1", "One");
        hMap.put("2", "Two");
        hMap.put("3", "Three");

        //create new Hashtable
        Hashtable ht = new Hashtable();

        //populate Hashtable
        ht.put("1", "This value would be REPLACED !!");
        ht.put("4", "Four");

        //print values of Hashtable before copy from HashMap
        System.out.println("Hashtable contents before copy");
        Enumeration e = ht.elements();
        while(e.hasMoreElements())
            System.out.println(e.nextElement());

        ht.putAll(hMap);

        //display contents of Hashtable
        System.out.println("Hashtable contents after copy");
        e = ht.elements();
        while(e.hasMoreElements())
            System.out.println(e.nextElement());
    }
}
```

Percobaan 7 : Mendapatkan key terendah dan tertinggi dari objek TreeMap.

```
import java.util.TreeMap;

public class GetLowestHighestKeyTreeMapExample {

    public static void main(String[] args) {
```

```

//create TreeMap object
TreeMap treeMap = new TreeMap();

//add key value pairs to TreeMap
treeMap.put("1","One");
treeMap.put("3","Three");
treeMap.put("2","Two");
treeMap.put("5","Five");
treeMap.put("4","Four");

System.out.println("Lowest key Stored in Java TreeMap is : "
                    + treeMap.firstKey());

System.out.println("Highest key Stored in Java TreeMap is : "
                    +
treeMap.lastKey());
}
}

```

Percobaan 8 : Mendapatkan TailMap dari objek TreeMap

```

import java.util.SortedMap;
import java.util.TreeMap;

public class GetTailMapFromTreeMapExample {

    public static void main(String[] args) {

        //create TreeMap object
        TreeMap treeMap = new TreeMap();

        //add key value pairs to TreeMap
        treeMap.put("1","One");
        treeMap.put("3","Three");
        treeMap.put("2","Two");
        treeMap.put("5","Five");
        treeMap.put("4","Four");

        SortedMap sortedMap = treeMap.tailMap("2");

        System.out.println("Tail Map Contains : " + sortedMap);
    }
}

```

Percobaan 9 : Mendapatkan SubMap dari objek TreeMap

```

import java.util.TreeMap;
import java.util.SortedMap;

public class GetSubMapFromTreeMapExample {

    public static void main(String[] args) {

```

```

//create TreeMap object
TreeMap treeMap = new TreeMap();

//add key value pairs to TreeMap
treeMap.put("1", "One");
treeMap.put("3", "Three");
treeMap.put("2", "Two");
treeMap.put("5", "Five");
treeMap.put("4", "Four");

SortedMap sortedMap = treeMap.subMap("2", "5");
System.out.println("SortedMap Contains : " + sortedMap);
}
}

```

Percobaan 10 : Mendapatkan HeadMap dari objek TreeMap

```

import java.util.SortedMap;
import java.util.TreeMap;

public class GetHeadMapFromTreeMapExample {

    public static void main(String[] args) {

        //create TreeMap object
        TreeMap treeMap = new TreeMap();

        //add key value pairs to TreeMap
        treeMap.put("1", "One");
        treeMap.put("3", "Three");
        treeMap.put("2", "Two");
        treeMap.put("5", "Five");
        treeMap.put("4", "Four");

        SortedMap sortedMap = treeMap.headMap("3");
        System.out.println("Head Map Contains : " + sortedMap);
    }
}

```

E. LATIHAN

Latihan 1 : Mengetahui penggunaan class TreeMap

Inputkan kalimat, buatlah sebagian kata-kata dalam kalimat tersebut ada yang sama, output berupa kata (sebagai key) dan jumlah kata (value) dalam kalimat tersebut yang tersimpan dalam TreeMap, selanjutnya tampilkan.

Input : televisi kursi televisi kursi meja televisi monitor.

Output : kursi = 2 meja = 1 monitor = 1 televisi = 3

Latihan 2 : Mengetahui penggunaan class TreeMap

Melanjutkan latihan 1, tampilkan :

- Tampilkan nilai terendah dan tertinggi

```
Output :  
Nilai terendah : Meja = 1  
Nilai tertinggi : Televisi = 3
```

- Tampilkan berdasarkan key dengan awalan m.

```
Output :  
meja = 1 monitor = 1
```

F. TUGAS

Tugas 1 : Ibukota propinsi di Indonesia

Terdapat objek TreeMap 1 yang berisi pulau(sebagai key) beserta propinsi-propinsinya(value). Terdapat objek TreeMap 2 yang berisi propinsi(sebagai key) beserta ibukotanya(value). Tampilkan :

- Ibukota propinsi yang terdapat di pulau Sumatera
- Ibukota propinsi yang terdapat di pulau Jawa
- Ibukota propinsi yang berawalan S (Sumatera Utara, Sumatera Barat, Sumatera Selatan, Sulawesi Barat, Sulawesi Tengah, Sulawesi Utara, Sulawesi Tenggara, Sulawesi Selatan)

G. LAPORAN RESMI

Kerjakan hasil percobaan(D), latihan(E) dan tugas(F) di atas dan tambahkan analisa.