# PRAKTIKUM 3

# JAVA COLLECTION FRAMEWORK

## A. TUJUAN PEMBELAJARAN

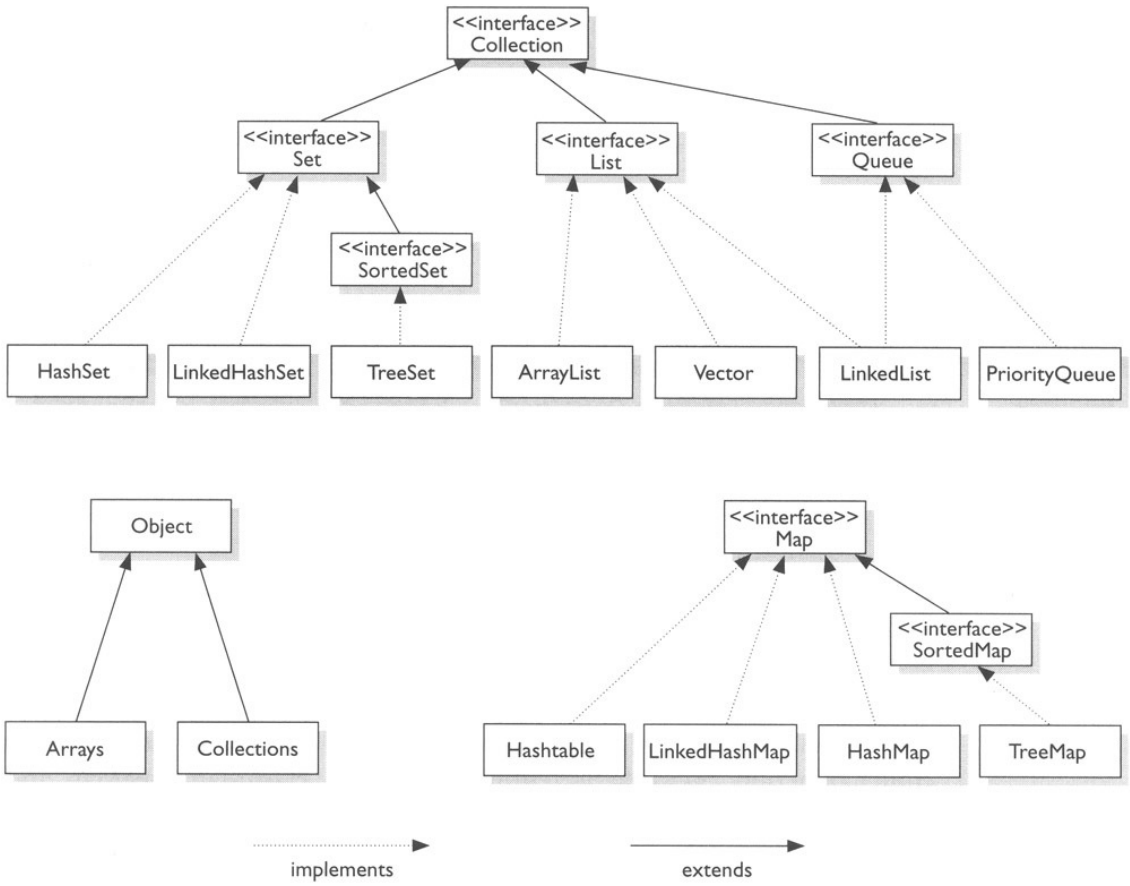
1. Memahami cara penyimpanan obyek menggunakan Collection.
2. Mengetahui pengelompokan dari Collection.
3. Mengetahui perbedaan dari interface Set, List dan Map.
4. Mengetahui penggunaan class-class dari interface Set, List dan Map.
5. Mengetahui cara penggunaan Iterasi dan Enumeration.

## B. DASAR TEORI

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




Gambar 3.1 Struktur Collection di Java

Java Collections Framework terbagi menjadi tiga kelompok:

- **Set**

Set mengikuti model himpunan, dimana obyek/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 clas TreeSet.

- **List**

List digunakan untuk menyimpan sekumpulan obyek berdasarkan urutan masuk (ordered) dan menerima duplikat. Cara penyimpanannya seperti array, oleh sebab itu memiliki posisi awal dan posisi akhir, menyisipkan obyek 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 obyek menggunakan metode FIFO (First In First Out) yaitu obyek yang masuk pertama keluar pertama. Class-class yang mengimplementasikan interface Queue adalah `PriorityQueue` dan `LinkedList`. Data yang tersimpan pada obyek `PriorityQueue` akan diurutkan, data tersebut harus mengimplementasikan obyek Comparable atau Comparator.

- **Map**

Perbedaaan mendasar map dengan collection yang lain, untuk menyimpan obyek pada Map, perlu sepasang obyek, 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 Java Collection Framework dan pembagian kelompok Collection.

## D. PERCOBAAN

**Percobaan 1 : Memahami penggunaan class-class yang mengimplementasikan interface Set yaitu class HashSet dan class TreeSet**

```
import java.util.*;
public class SetExample {
    public static void main(String[] args) {
        Set set=new HashSet();
        set.add("Bernadine");
        set.add("Elizabeth");
```

```
        set.add("Gene");
        set.add("Elizabeth");
        set.add("Clara");
        System.out.print("Elemen pada HashSet : ");
        System.out.println(set);
        Set sortSet=new TreeSet(set);
        System.out.print("Elemen pada TreeSet : ");
        System.out.println(sortSet);
    }
}
```

**Percobaan 2 : Penggunaan Class HashSet**

```
public class FindDups {
    public static void main(String[] args) {
        Set<String> s = new HashSet<String>();
        for (String a : args)
            if (!s.add(a))
                System.out.println("Duplicate detected: " + a);

        System.out.println(s.size() + " distinct words: " + s);
    }
}
```

**Jalankan dengan :**

**`java FindDups i came i saw i left`**

```
public class FindDups2 {
    public static void main(String[] args) {
        Set<String> uniques = new HashSet<String>();
        Set<String> dups    = new HashSet<String>();

        for (String a : args)
            if (!uniques.add(a))
                dups.add(a);

        // Destructive set-difference
        uniques.removeAll(dups);

        System.out.println("Unique words:    " + uniques);
        System.out.println("Duplicate words: " + dups);
    }
}
```

**Jalankan dengan**

**`java FindDups i came i saw i left`**

**Percobaan 3 : Interface Set menerapkan konsep himpunan. Mengetahui implementasi konsep himpunan pada interface Set.**

```
import java.util.*;
public class SetExample {
    public static void main(String[] args) {
        Set s1=new HashSet();
        s1.add("Australia");
        s1.add("Sweden");
        s1.add("Germany");

        Set s2=new HashSet();
        s2.add("Sweden");
        s2.add("France");

        Set union=new TreeSet(s1);
        union.addAll(s2); // gabungan dari s1 dan s2
        print("Union",union);

        Set intersect=new TreeSet(s1);
        intersect.retainAll(s2); // irisan dari s1 dan s2
        print("Intersection",intersect);
    }
    protected static void print(String label, Collection c){
        System.out.println("---------------- "+ label+" ---------------
");

        Iterator it=c.iterator();
        while(it.hasNext()){
            System.out.println(it.next());
        }
    }
}
```

**Percobaan 4 : Memahami penggunaan class-class yang mengimplementasikan interface List yaitu ArrayList dan LinkedList.**

```
import java.util.*;

public class ListExample {
    public static void main(String[] args) {
        List list=new ArrayList();
        list.add("Bernadine");
        list.add("Elizabeth");
        list.add("Gene");
        list.add("Elizabeth");
        list.add("Clara");

        System.out.println(list);

        System.out.println("2 : "+list.get(2));
        System.out.println("0 : "+list.get(0));

        LinkedList queue=new LinkedList();
        queue.addFirst("Bernadine");
        queue.addFirst("Elizabeth");
```

```
        queue.addFirst("Gene");
        queue.addFirst("Elizabeth");
        queue.addFirst("Clara");

        System.out.println(queue);
        queue.removeLast();
        queue.removeLast();
        System.out.println(queue);
    }
}
```

**Percobaan 5 : Penggunaan Class Vector**

```
import java.util.Vector;

public class VectorExample {

    public static void main(String[] args) {

        Vector<String> vc=new Vector<String>();

        //<E> Element type of Vector e.g. String, Integer, Object ...

        // add vector elements
        vc.add("Vector Object 1");
        vc.add("Vector Object 2");
        vc.add("Vector Object 3");
        vc.add("Vector Object 4");
        vc.add("Vector Object 5");

        // add vector element at index
        vc.add(3, "Element at fix position");

        // vc.size() inform number of elements in Vector
        System.out.println("Vector Size :"+vc.size());

        // get elements of Vector
        for(int i=0;i<vc.size();i++)
        {
            System.out.println("Vector Element "+i+" :"+vc.get(i));
        }
    }
}
```

**Percobaan 6 : Penggunaan Iterator**

```
import java.util.*;

class IteratorDemo {

    public static void main(String args[]) {
        // create an array list
```

```
        ArrayList al = new ArrayList();
        // add elements to the array list
        al.add("C");
        al.add("A");
        al.add("E");
        al.add("B");
        al.add("D");
        al.add("F");
        // use iterator to display contents of al
        System.out.print("Original contents of al: ");
        Iterator itr = al.iterator();
        while (itr.hasNext()) {

            Object element = itr.next();
            System.out.print(element + " ");

        }
        System.out.println();
        // modify objects being iterated
        ListIterator litr = al.listIterator();
        while (litr.hasNext()) {

            Object element = litr.next();
            litr.set(element + "+");

        }
        System.out.print("Modified contents of al: ");
        itr = al.iterator();
        while (itr.hasNext()) {

            Object element = itr.next();
            System.out.print(element + " ");

        }
        System.out.println();
        // now, display the list backwards
        System.out.print("Modified list backwards: ");
        while (litr.hasPrevious()) {

            Object element = litr.previous();
            System.out.print(element + " ");

        }
        System.out.println();
    }
}
```

**Percobaan 7 : Penggunaan Enumeration**

```
import java.util.Vector;
import java.util.Enumeration;

public class EnumerationTester {
```

```
    public static void main(String args[]) {
        Enumeration days;
        Vector dayNames = new Vector();
        dayNames.add("Sunday");
        dayNames.add("Monday");
        dayNames.add("Tuesday");
        dayNames.add("Wednesday");
        dayNames.add("Thursday");
        dayNames.add("Friday");
        dayNames.add("Saturday");
        days  = dayNames.elements();

         while (days.hasMoreElements ())
            System.out.println(days.nextElement());
    }
}
```

**Percobaan 8 : Membuat Array List dari Enumerasi**

```
public class CreateArrayListFromEnumerationExample {

  public static void main(String[] args) {

    //create a Vector object
    Vector v = new Vector();

    //Add elements to Vector
    v.add("A");
    v.add("B");
    v.add("D");
    v.add("E");
    v.add("F");
    System.out.println("Vector contains : " + v);

    //Get Enumeration over Vector

    Enumeration e = v.elements();
    //Create ArrayList from Enumeration of Vector
    ArrayList aList = Collections.list(e);
    System.out.println("Arraylist contains : " + aList);
  }
}
```

**Percobaan 9 : Mengkopikan element dari ArrayList ke Vector**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Vector;

public class CopyElementsOfArrayListToVectorExample {
```

```
  public static void main(String[] args) {
    //create an ArrayList object
    ArrayList arrayList = new ArrayList();

    //Add elements to Arraylist
    arrayList.add("1");
    arrayList.add("4");
    arrayList.add("2");
    arrayList.add("5");
    arrayList.add("3");

    //create a Vector object
    Vector v = new Vector();

    //Add elements to Vector
    v.add("A");
    v.add("B");
    v.add("D");
    v.add("E");
    v.add("F");
    v.add("G");
    v.add("H");

    System.out.println("Before copy, Vector Contains : " + v);

    //copy all elements of ArrayList to Vector using copy method of
Collections class

    Collections.copy(v,arrayList);

    System.out.println("After Copy, Vector Contains : " + v);
  }
}
```

**Percobaan 10 : Menambahkan elemen yang tersimpan di Collection pada ArrayList**

```
import java.util.ArrayList;
import java.util.Vector;

public class AppendAllElementsOfOtherCollectionToArrayListExample
{

    public static void main(String[] args) {

        //create an ArrayList object
        ArrayList arrayList = new ArrayList();

        //Add elements to Arraylist
        arrayList.add("1");
        arrayList.add("2");
        arrayList.add("3");
```

```
        //create a new Vector object
        Vector v = new Vector();
        v.add("4");
        v.add("5");

        //append all elements of Vector to ArrayList
        arrayList.addAll(v);

        //display elements of ArrayList

        System.out.println("After appending all elements of Vector,
                            ArrayList contains..");

    for (int i = 0; i < arrayList.size(); i++) {
            System.out.println(arrayList.get(i));
        }
    }
}
```

**Percobaan 11 : Memahami Penggunaan dari class PriorityQueue**

```
import java.util.*;

public class PriorityQueueDemo {
   PriorityQueue<String> stringQueue;
   public static void main(String[] args){
      stringQueue = new PriorityQueue<String>();
      stringQueue.add("ab");
      stringQueue.add("abcd");
      stringQueue.add("abc");
      stringQueue.add("a");

      //don't use iterator which may or may not
      //show the PriorityQueue's order
      while(stringQueue.size() > 0)
       System.out.println(stringQueue.remove());
      }
}
```

**Percobaan 12 : Memahami Penggunaan dari class PriorityQueue dan data yang tersimpan dalam obyek PriorityQueue mengimplementasikan interface Comparator.**

```
import java.util.Comparator;
import java.util.PriorityQueue;

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

```
        PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>(10,
new Comparator<Integer>() {

            public int compare(Integer int1, Integer int2) {
                boolean flag1 = isPrime(int1);
                boolean flag2 = isPrime(int2);

                if (flag1 == flag2){
                    return int1.compareTo(int2);
                } else if (flag1) {
                    return -1;
                } else if(flag2) {
                    return 1;
                }
                return 0;
            }
        });

        pQueue.add(1);
        pQueue.add(5);
        pQueue.add(6);
        pQueue.add(4);
        pQueue.add(2);
        pQueue.add(9);
        pQueue.add(7);
        pQueue.add(8);
        pQueue.add(10);
        pQueue.add(3);

        while(true) {
            Integer head = pQueue.poll();
            if(head == null) {
                break;
            }
            System.out.print(head + "  <-- ");
        }
    }

    /**
     *
     * @param n
     * @return
     */
    public static boolean isPrime(int n) {
        if (n <= 1) {
            return false;
        }
        if (n == 2) {
            return true;
        }
        if (n % 2 == 0) {
            return false;
        }
        long m = (long) Math.sqrt(n);
```

```
        for (long i = 3; i  <= m; i += 2) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

**Percobaan 13 : 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 14 : 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 15 : 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 16 : 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 17 : 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 18 : 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 19 : 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 20 : 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 21 : 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 22 : 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 : Penerapan konsep himpunan pada interface Set

Terdapat sebuah himpunan

A = {1,2,3,4,5}

B = {5,6,7,8,9,10}

Menggunakan class yang mengimplementasikan Interface Set, dapatkah output seperti :

– A – B

– A ∩ B

– A U B

– A C B

### Latihan 2 : Memahami penggunaan interface List.

Buatlah obyek List, dengan data bertipe String lakukan langkah berikut :

- Tampilkan data yang terdapat pada list.
- Baliklah data yang terdapat pada list dan tampilkan.
- Acaklah data tersebut dan tampilkan.
- Urutkan data tersebut dan tampilkan.

**Latihan 3 : Memahami penggunaan interface List (2)**

Buatlah class Mahasiswa dengan informasi nrp dan nama(bertipe String).

Buatlah obyek List, dengan data bertipe String lakukan langkah berikut :

- Tampilkan data yang terdapat pada list.
- Baliklah data yang terdapat pada list dan tampilkan.
- Acaklah data tersebut dan tampilkan.
- Urutkan data tersebut, jangan lupa untuk mengimplementasikan interface Comparable/Comparator pada class Mahasiswa dan tampilkan.

**Latihan 4 : Penggunaan class LinkedList pada interface List.**

Buatlah dua obyek List (ArrayList) yaitu obyek warna dan warnaDihapus. Obyek ini berisi warna-warna, buatlah sebagian ada yang sama. Lakukan penghapusan data yang terdapat pada obyek warna yang sama dengan data warna yang terdapat pada obyek warnaDihapus, selanjutnya tampilkan.

```
Warna :
[MAGENTA, RED, WHITE, BLUE, CYAN]
Warna yang dihapus :
[RED, WHITE, BLUE]

Output :
Warna :
[MAGENTA, CYAN]
```

**Latihan 5 : Pengurutan data mahasiswa berdasarkan nilai.**

Buatlah class Mahasiswa dengan informasi :

- Nrp (String)
- Nama(String)
- Nilai(Float)

Terdapat 10 data mahasiswa yang tersimpan dalam queue, set nilai secara random antara 60-100, lakukan pengurutan data mahasiswa berdasarkan nilai tersebut !

**Latihan 6 : 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.<br>Output : kursi = 2 meja = 1 monitor = 1 televisi = 3 |

**Latihan 7 : 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
```

**Latihan 8 : 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)

## F. LAPORAN RESMI

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