PRAKTIKUM 4

COMPARABLE DAN COMPARATOR

A. TUJUAN PEMBELAJARAN

1. Mengetahui untuk mengurutkan data dengan cara membandingkan satu objek dengan objek lainnya.
2. Mengetahui class-class di Java yang mengimplementasikan interface Comparable.
3. Mengetahui cara mengurutkan data dengan class yang didefinisikan sendiri, dengan cara mengimplementasikan interface Compable dan Comparator.

B. DASAR TEORI

Pada kehidupan nyata, object-object sering dibandingkan, misal :

- Mobil Andi lebih mahal dibandingkan dengan mobil Budi
- Buku A lebih tebal dibandingkan dengan Buku B
- Usia Andi lebih muda dibandingkan dengan usia Intan

Dalam pemrograman object oriented, sering sekali ada kebutuhan untuk membandingkan object-object dari class yang sama, misalkan membandingkan object untuk mengurutkan data, pencarian data yang diurutkan berdasarkan umur. Pada praktikum ini akan membahas bagaimana merancang object dari class untuk bisa dibandingkan menggunakan interface java.lang.Comparable and java.util.Comparator.

B.1 Mengurutkan Obyek String

Terdapat array dengan tipe String, untuk mengurutkan data String pada array gunakan Arrays.sort().

```
import java.util.Arrays;

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

```
        String animals[] = new String[6];
        animals[0] = "snake";
        animals[1] = "kangaroo";
        animals[2] = "wombat";
        animals[3] = "bird";

        System.out.println("\nSEBELUM DISORTING");
        for (int i = 0; i < 4; i++) {
            System.out.println("animal " + i + " : " + animals[i]);
        }

        Arrays.sort(animals,0,4);
        System.out.println("\nSETELAH DISORTING");
        for (int i = 0; i < 4; i++) {
            System.out.println("animal " + i + " : " + animals[i]);
        }
    }
}
```

Output :

```
SEBELUM DISORTING
animal 0 : snake
animal 1 : kangaroo
animal 2 : wombat
animal 3 : bird

SETELAH DISORTING
animal 0 : bird
animal 1 : kangaroo
animal 2 : snake
animal 3 : wombat
```

Terdapat data String yang tersimpan dalam ArrayList, untuk mengurutkan data menggunakan Collections.sort()

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

public class SortList {
    public static void main(String args[]){
        ArrayList insects = new ArrayList();
        insects.add("mosquito");
        insects.add("butterfly");
        insects.add("dragonfly");
        insects.add("fly");

        System.out.println("\nSEBELUM DISORTING");
        int size = insects.size();
        for (int i = 0; i < size; i++) {
```

```
            System.out.println("insect " + i + " : " + (String)
insects.get(i));
        }

        Collections.sort(insects);

        System.out.println("\nSETELAH DISORTING");
        for (int i = 0; i < size; i++) {
            System.out.println("insect " + i + " : " + (String)
insects.get(i));
        }
    }
}
```

Output :

```
SEBELUM DISORTING
insect 0 : mosquito
insect 1 : butterfly
insect 2 : dragonfly
insect 3 : fly

SETELAH DISORTING
insect 0 : butterfly
insect 1 : dragonfly
insect 2 : fly
insect 3 : mosquito
```

B.2 Mengurutkan Objek Yang Kita Definisikan Sendiri

Kita buat class Mahasiswa yang terdapat informasi nama dan nrp dengan tipe String. Terdapat beberapa data mahasiswa yang disimpan di array, selanjutnya kita urutkan berdasarkan nrp.

```
public class Mahasiswa {
    private String nrp ;
    private String nama ;

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

    public String getNama() {
        return nama;
    }

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

```
    public String getNrp() {
        return nrp;
    }

    public void setNrp(String nrp) {
        this.nrp = nrp;
    }

    @Override
    public String toString() {
        return "Mahasiswa{" + "nrp=" + nrp + " nama=" + nama + '}';
    }
}
```

Buatlah class TestMahasiswa untuk menampilkan output :

```
import java.util.Arrays;

public class TestMahasiswa {
    public static void main(String[] args) {
        Mahasiswa dataMhs[] = {new Mahasiswa("05", "Cahya"),new
Mahasiswa("04", "Rudi"),
        new Mahasiswa("01", "Endah"),new Mahasiswa("03", "Rita"),new
Mahasiswa("02", "Tika")};
        System.out.println("SEBELUM SORTING");
        System.out.println(Arrays.toString(dataMhs));
        Arrays.sort(dataMhs);
        System.out.println("\nSESUDAH SORTING");
        System.out.println(Arrays.toString(dataMhs));
    }
}
```

Output :

```
Exception in thread "main" java.lang.ClassCastException: prak.Mahasiswa
cannot be cast to java.lang.Comparable
        at java.util.Arrays.mergeSort(Arrays.java:1144)
        at java.util.Arrays.sort(Arrays.java:1079)
        at prak.TestMahasiswa.main(TestMahasiswa.java:16)
Java Result: 1
```

Pada output program terjadi exception. Mengapa? Karena untuk mengurutkan objek (proses mengurutkan ini dilakukan dengan cara membandingkan satu objek dengan objek yang lain), maka objek tersebut harus mengimplementasikan interface Comparable. Dengan mengimplementasikan interface Comparable pada sebuah class, menyebabkan object-object tersebut bisa dibandingkan (comparable).

Kalau pada percobaan sebelumnya data yang diurutkan adalah String, dapat diurutkan karena String mengimplementasikan interface Comparable.

```
public final class String extends Object implements Serializable,
Comparable<String>, CharSequence
```

Interface ini mempunyai sebuah method `compareTo()` yang menentukan bagaimana cara membandingkan antara dua object dari class tersebut.

Bentuk methodnya:

```
public int compareTo(Object o)
```

Mehod `compareTo()` menerima Object, sehingga kita bisa memasukkan sembarang object, tapi harus mempunyai tipe yang sama. Kalau object yang kita masukkan adalah object yang berbeda maka melemparkan exception java.lang.ClassCastException. Return value dari method compareTo()

- 0 jika dua object yang dibandingkan sama.
- Bilangan positif, jika object 1 lebih besar dibandingkan dengan object 2
- Bilangan negatif, jika object 1 lebih kecil dibandingkan dengan object 2

Bagaimana caranya supaya bisa menggunakan Array.sort() pada contoh kasus diatas. Pada class Mahasiswa implementasikan interface Comparable, berarti harus mengimplementasikan method compareTo(). Isilah method ini dengan tujuan untuk membandingkan object dari class Mahasiswa berdasarkan umur. Jangan lupa untuk mengcasting object menjadi object dari class Mahasiswa terlebih dahulu.

```
public class Mahasiswa implements Comparable {
    private String nrp ;
    private String nama ;

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

    public String getNama() {
        return nama;
    }

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

```
    public String getNrp() {
        return nrp;
    }

    public void setNrp(String nrp) {
        this.nrp = nrp;
    }

    @Override
    public String toString() {
        return "Mahasiswa{" + "nrp=" + nrp + " nama=" + nama + '}';
    }

    public int compareTo(Object o) {
        Mahasiswa m2 = (Mahasiswa) o ;
        return this.nrp.compareTo(m2.nrp);
    }

}
```

Selanjutnya jalankan class TestMahasiswa lagi.

Output :

```
SEBELUM SORTING
[Mahasiswa{nrp=05 nama=Cahya}, Mahasiswa{nrp=04 nama=Rudi},
Mahasiswa{nrp=01 nama=Endah}, Mahasiswa{nrp=03 nama=Rita},
Mahasiswa{nrp=02 nama=Tika}]

SESUDAH SORTING
[Mahasiswa{nrp=01 nama=Endah}, Mahasiswa{nrp=02 nama=Tika},
Mahasiswa{nrp=03 nama=Rita}, Mahasiswa{nrp=04 nama=Rudi},
Mahasiswa{nrp=05 nama=Cahya}]
```

B.3 Penggunaan Class Comparator

Dengan mengimplementasikan interface `Comparable` kita hanya bisa menentukan satu cara saja untuk membandingkan object-object dari class `Mahasiswa`, untuk contoh sebelumnya, yang kita bandingkan berdasarkan nrp. Bagaimana jika object-object dari class `Mahasiswa` diurutkan berdasarkan nama? Berarti object-object tersebut dibandingkan berdasarkan nama. Kita masih memerlukan satu cara lagi untuk membandingkan object-object dari class `Mahasiswa`. Kita memerlukan comparator. Untuk membuat comparator, buat class yang mengimplementasikan interface java.util.Comparator, dan method `compare()`.

```
public int compare(Object o1, Object o2)
```

Return value dari method compare()

- 0 jika dua object yang dibandingkan sama.
- Bilangan positif, jika object 1 lebih besar dibandingkan dengan object 2
- Bilangan negatif, jika object 1 lebih kecil dibandingkan dengan object 2

Class Comparator

```
public class NamaComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        Mahasiswa m1 = (Mahasiswa) o1;
        Mahasiswa m2 = (Mahasiswa) o2;
        return m1.getNama().compareTo(m2.getNama());
    }
}
```

Penggunaan objek Comparator pada Arrays.sort()

```
public class TestMahasiswa2 {
    public static void main(String[] args) {
        Mahasiswa dataMhs[] = {new Mahasiswa("05", "Cahya"),new
Mahasiswa("04", "Rudi"),
        new Mahasiswa("01", "Endah"),new Mahasiswa("03", "Rita"),new
Mahasiswa("02", "Tika")};

        System.out.println("SEBELUM SORTING");
        System.out.println(Arrays.toString(dataMhs));
        Arrays.sort(dataMhs, new NamaComparator());
        System.out.println("\nSESUDAH SORTING");
        System.out.println(Arrays.toString(dataMhs));
    }
}
```

Output : Mengurutkan Data Mahasiswa berdasarkan Nama

```
SEBELUM SORTING
[Mahasiswa{nrp=05 nama=Cahya}, Mahasiswa{nrp=04 nama=Rudi},
Mahasiswa{nrp=01 nama=Endah}, Mahasiswa{nrp=03 nama=Rita},
Mahasiswa{nrp=02 nama=Tika}]

SESUDAH SORTING
[Mahasiswa{nrp=05 nama=Cahya}, Mahasiswa{nrp=01 nama=Endah},
Mahasiswa{nrp=03 nama=Rita}, Mahasiswa{nrp=04 nama=Rudi},
Mahasiswa{nrp=02 nama=Tika}]
```

C. TUGAS PENDAHULUAN

Buatlah resume 1 halaman mengenai interface Comparable dan berikan 1 contoh penggunaan interface Comparable dan Comparator

D. PERCOBAAN

Percobaan 1 : Mengurutkan data dengan tipe String yang tersimpan di array.

```
import java.util.Arrays;

public class ArrayString {
    public static void main(String args[]){
        String animals[] = new String[6];
        animals[0] = "snake";
        animals[1] = "kangaroo";
        animals[2] = "wombat";
        animals[3] = "bird";

        System.out.println("\nSEBELUM DISORTING");
        for (int i = 0; i < 4; i++) {
            System.out.println("animal " + i + " : " + animals[i]);
        }

        Arrays.sort(animals,0,4);
        System.out.println("\nSETELAH DISORTING");
        for (int i = 0; i < 4; i++) {
            System.out.println("animal " + i + " : " + animals[i]);
        }
    }
}
```

Percobaan 2 : Mengurutkan data dengan tipe String yang tersimpan di List.

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

public class SortList {
    public static void main(String args[]){
        ArrayList insects = new ArrayList();
        insects.add("mosquito");
        insects.add("butterfly");
        insects.add("dragonfly");
        insects.add("fly");

        System.out.println("\nSEBELUM DISORTING");
        int size = insects.size();
        for (int i = 0; i < size; i++) {
            System.out.println("insect " + i + " : " + (String)
insects.get(i));
        }

        Collections.sort(insects);
```

```
        System.out.println("\nSETELAH DISORTING");
        for (int i = 0; i < size; i++) {
            System.out.println("insect " + i + " : " + (String)
insects.get(i));
        }
    }
}
```

Percobaan 3 : Membuat class Mahasiswa dengan variabel nama dan nrp dengan tipe String. Membuat data mahasiswa yang tersimpan di array. Selanjutnya lakukan pengurutan data mahasiswa tersebut, apa yang terjadi? Jelaskan!

```
public class Mahasiswa {
    private String nrp ;
    private String nama ;

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

    public String getNama() {
        return nama;
    }

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

    public String getNrp() {
        return nrp;
    }

    public void setNrp(String nrp) {
        this.nrp = nrp;
    }

    @Override
    public String toString() {
        return "Mahasiswa{" + "nrp=" + nrp + " nama=" + nama + '}';
    }
}
```

```
import java.util.Arrays;

public class TestMahasiswa {
    public static void main(String[] args) {
        Mahasiswa dataMhs[] = {new Mahasiswa("05", "Cahya"),new
Mahasiswa("04", "Rudi"),
```

```
        new Mahasiswa("01", "Endah"),new Mahasiswa("03", "Rita"),new
Mahasiswa("02", "Tika")};
        System.out.println("SEBELUM SORTING");
        System.out.println(Arrays.toString(dataMhs));
        Arrays.sort(dataMhs);
        System.out.println("\nSESUDAH SORTING");
        System.out.println(Arrays.toString(dataMhs));
    }
}
```

Percobaan 4 : Mengimplementasikan interface Comparable pada class Mahasiswa untuk membandingkan antar objek mahasiswa berdasarkan nrp, lalu urutkan data mahasiswa, apa yang terjadi ? Jelaskan !

```
public class Mahasiswa implements Comparable {
    private String nrp ;
    private String nama ;

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

    public String getNama() {
        return nama;
    }

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

    public String getNrp() {
        return nrp;
    }

    public void setNrp(String nrp) {
        this.nrp = nrp;
    }

    @Override
    public String toString() {
        return "Mahasiswa{" + "nrp=" + nrp + " nama=" + nama + '}';
    }

    public int compareTo(Object o) {
        Mahasiswa m2 = (Mahasiswa) o ;
        return this.nrp.compareTo(m2.nrp);
    }

}
```

Percobaan 5 : Mengurutkan data mahasiswa berdasarkan nama, dengan mengimplementasikan interface Comparator.

Class Comparator

```
public class NamaComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        Mahasiswa m1 = (Mahasiswa) o1;
        Mahasiswa m2 = (Mahasiswa) o2;
        return m1.getNama().compareTo(m2.getNama());
    }
}
```

Penggunaan objek Comparator pada Arrays.sort()

```
public class TestMahasiswa2 {
    public static void main(String[] args) {
        Mahasiswa dataMhs[] = {new Mahasiswa("05", "Cahya"),new
Mahasiswa("04", "Rudi"),
        new Mahasiswa("01", "Endah"),new Mahasiswa("03", "Rita"),new
Mahasiswa("02", "Tika")};

        System.out.println("SEBELUM SORTING");
        System.out.println(Arrays.toString(dataMhs));
        Arrays.sort(dataMhs, new NamaComparator());
        System.out.println("\nSESUDAH SORTING");
        System.out.println(Arrays.toString(dataMhs));
    }
}
```

Percobaan 6 : Mengurutkan data secara descending, data tersimpan di ArrayList

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

public class SortArrayListInDescendingOrderExample {

  public static void main(String[] args) {

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

    //Add elements to Arraylist
    arrayList.add("A");
    arrayList.add("B");
    arrayList.add("C");
    arrayList.add("D");
    arrayList.add("E");

    Comparator comparator = Collections.reverseOrder();
```

```
    System.out.println("Before sorting ArrayList in descending order :
"                                                                    +
arrayList);

    Collections.sort(arrayList,comparator);
    System.out.println("After sorting ArrayList in descending order : "
                                                                     +
arrayList);
  }
}
```

Percobaan 7 : Membuat sendiri class yang akan diurutkan dan membuat objek Comparator.

```
import java.util.*;

class Employee{

    private int age;
    private String name;

    public void setAge(int age){
        this.age=age;
    }

    public int getAge(){
        return this.age;
    }

    public void setName(String name){
        this.name=name;
    }

    public String getName(){
        return this.name;
    }
}

class AgeComparator implements Comparator{

    public int compare(Object emp1, Object emp2){

        int emp1Age = ((Employee)emp1).getAge();
        int emp2Age = ((Employee)emp2).getAge();

        if(emp1Age > emp2Age)
            return 1;
        else if(emp1Age < emp2Age)
            return -1;
        else
            return 0;
```

```
    }
}

class NameComparator implements Comparator{

    public int compare(Object emp1, Object emp2){

        //parameter are of type Object, so we have to downcast it to
Employee objects

        String emp1Name = ((Employee)emp1).getName();
        String emp2Name = ((Employee)emp2).getName();

        //uses compareTo method of String class to compare names of the
employee
        return emp1Name.compareTo(emp2Name);
    }
}

public class JavaComparatorExample{

    public static void main(String args[]){

        //Employee array which will hold employees
        Employee employee[] = new Employee[2];

        //set different attributes of the individual employee.
        employee[0] = new Employee();
        employee[0].setAge(40);
        employee[0].setName("Joe");

        employee[1] = new Employee();
        employee[1].setAge(20);
        employee[1].setName("Mark");

        System.out.println("Order of employee before sorting is");
        //print array as is.
        for(int i=0; i < employee.length; i++){
            System.out.println( "Employee " + (i+1) + " name :: " +
employee[i].getName() + ", Age :: " + employee[i].getAge());
        }

        //Sorting array on the basis of employee age by passing
AgeComparator
        Arrays.sort(employee, new AgeComparator());
        System.out.println("\n\nOrder of employee after sorting by
employee age is");

        for(int i=0; i < employee.length; i++){
            System.out.println( "Employee " + (i+1) + " name :: " +
employee[i].getName() + ", Age :: " + employee[i].getAge());
        }
```

```
        //Sorting array on the basis of employee Name by passing
NameComparator
        Arrays.sort(employee, new NameComparator());

        System.out.println("\n\nOrder of employee after sorting by
employee name is");
        for(int i=0; i < employee.length; i++){
            System.out.println( "Employee " + (i+1) + " name :: " +
employee[i].getName() + ", Age :: " + employee[i].getAge());
        }

    }

}
```

Percobaan 8 : Membuat sendiri class yang akan diurutkan dan membuat objek Comparator. Mengurutkan objek secara descending.

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class EmpComparator implements Comparator {
  public int compare(Object obj1, Object obj2) {
    Person emp1 = (Person) obj1;
    Person emp2 = (Person) obj2;

    int nameComp = emp1.getFirstName().compareTo(emp2.getFirstName());

    return ((nameComp == 0) ? emp1.getLastName().compareTo(
        emp2.getLastName()) : nameComp);
  }

  public static void main(String args[]) {
    String names[] = { "Bart", "Hugo", "Lisa", "Marge", "Homer",
"Maggie",
        "Roy" };

    // Convert to list
    List list = new ArrayList(Arrays.asList(names));

    // Ensure list sorted
    Collections.sort(list);
    System.out.println("Sorted list: [length: " + list.size() + "]");
    System.out.println(list);

    // Search for element in list
    int index = Collections.binarySearch(list, "Maggie");
    System.out.println("Found Maggie @ " + index);
```

```
    // Search for element not in list
    index = Collections.binarySearch(list, "Jimbo Jones");
    System.out.println("Didn't find Jimbo Jones @ " + index);

    // Insert
    int newIndex = -index - 1;
    list.add(newIndex, "Jimbo Jones");
    System.out.println("With Jimbo Jones added: [length: " +
list.size()
        + "]");
    System.out.println(list);

    System.out.println(Collections.min(list));
    System.out.println(Collections.max(list));
```

```
    Comparator comp = Collections.reverseOrder();

    System.out.println(Collections.min(list, comp));
     System.out.println(Collections.max(list, comp));
  }
}
```

```
class Person implements Comparable {
  String firstName, lastName;

  public Person(String f, String l) {
    this.firstName = f;
    this.lastName = l;
  }

  public String getFirstName() {
    return firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public String toString() {
    return "[ name=" + firstName + ",name=" + lastName + "]";
  }

  public int compareTo(Object obj) {
    Person emp = (Person) obj;
    int deptComp = firstName.compareTo(emp.getFirstName());

    return ((deptComp == 0) ? lastName.compareTo(emp.getLastName())
        : deptComp);
  }

  public boolean equals(Object obj) {
```

```
    if (!(obj instanceof Person)) {
      return false;
    }
    Person emp = (Person) obj;
    return firstName.equals(emp.getFirstName())
        && lastName.equals(emp.getLastName());
  }

}
```

E. LATIHAN

Latihan 1 : Sebutkan class-class yang mengimplementasikan interface Comparable.

Latihan 2 : Kembangkan untuk Class Mahasiswa dengan memberikan variabel baru berupa nilai IPK (double), selanjutkan lakukan pengurutan data Mahasiswa berdasarkan nrp, nama dan nilai IPK (menggunakan Comparable dan Comparator)

Latihan 3 : Ubahlah contoh FindDups di bawah ini menggunakan SortedSet untuk menggantikan Set. Definisikan Comparator, sehingga pada saat melakukan sorting berlaku non- case sensitive.

```
import java.util.*;

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);
    }
}
```

F. LAPORAN RESMI

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