MEMBUAT OBJECT COMPARABLE DAN COMPARATOR

Yuliana Setiowati
Politeknik Elektronika Negeri Surabaya

Membuat Object Comparable

- 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.
- Pertemuan ini akan membahas bagaimana merancang object dari class untuk bisa dibandingkan menggunakan interface java.lang.Comparable and java.util.Comparator

Mengurutkan Object 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]);
        }
    }
}
```

```
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
BUILD SUCCESSFUL (total time: 0 seconds)
```


Mengurutkan Object String

- 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));
        }
    }
}
```

```
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
```

Class Person

```
class Person {
  private String firstName;
  private String lastName;
  private int age;

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  public int getAge() {
    return age;
  }

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


Mengurutkan object Person

```
import java.util.Arrays;
public class TestPerson {
    public static void main(String args[]){
        Person[] persons = new Person[4];
        persons[0] = new Person();
        persons[0].setFirstName("Elvis");
        persons[0].setLastName("Goodyear");
        persons[0].setAge(56);

        persons[1] = new Person();
        persons[1].setFirstName("Stanley");
        persons[1].setLastName("Clark");
        persons[1].setAge(8);

        persons[2] = new Person();
        persons[2].setFirstName("Jane");
        persons[2].setLastName("Graff");
        persons[2].setAge(16);

        persons[3] = new Person();
        persons[3].setFirstName("Nancy");
        persons[3].setLastName("Goodyear");
        persons[3].setAge(69);
        Arrays.sort(persons);
    }
}
```

- Terdapat data-data (object) dari class Person yang disimpan dalam array Person. Data akan diurutkan menggunakan Array.sort(). Apa yang terjadi ?

Mengurutkan object Person

- Output program
- Program melempar ClassCastException

```
Exception in thread "main" java.lang.ClassCastException: Person cannot be cast to java.lang.Comparable
        at java.util.Arrays.mergeSort(Arrays.java:1144)
        at java.util.Arrays.sort(Arrays.java:1079)
        at TestPerson.main(TestPerson.java:29)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```


Menggunakan Interface java.lang.Comparable

- Dengan mengimplementasikan interface Comparable pada sebuah class, menyebabkan object-object tersebut bisa dibandingkan (comparable).
- 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 melempar 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

Menggunakan Interface java.lang.Comparable

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

```
public int compareTo(Object anotherPerson) throws ClassCastException {
  if (!(anotherPerson instanceof Person))
    throw new ClassCastException("A Person object expected.");
  int anotherPersonAge = ((Person) anotherPerson).getAge();
  return this.age - anotherPersonAge;
}
```


Menggunakan Interface java.lang.Comparable

```
class Person implements Comparable {
  private String firstName;
  private String lastName;
  private int age;

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }
```

Menggunakan Interface java.lang.Comparable

```
  public int getAge() {
    return age;
  }

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

  public int compareTo(Object anotherPerson) throws ClassCastException {
    if (!(anotherPerson instanceof Person))
      throw new ClassCastException("A Person object expected.");
    int anotherPersonAge = ((Person) anotherPerson).getAge();
    return this.age - anotherPersonAge;
  }
}
```


Class Testing

```
package comparable.ex01;
import java.util.Arrays;

public class Testing {
  //diurutkan berdasarkan umur
  public static void main(String[] args) {
    Person[] persons = new Person[4];
    persons[0] = new Person();
    persons[0].setFirstName("Elvis");
    persons[0].setLastName("Goodyear");
    persons[0].setAge(56);

    persons[1] = new Person();
    persons[1].setFirstName("Stanley");
    persons[1].setLastName("Clark");
    persons[1].setAge(8);

    persons[2] = new Person();
    persons[2].setFirstName("Jane");
    persons[2].setLastName("Graff");
    persons[2].setAge(16);

    persons[3] = new Person();
    persons[3].setFirstName("Nancy");
    persons[3].setLastName("Goodyear");
    persons[3].setAge(69);
```

Class Testing

```
System.out.println("Natural Order");

for (int i=0; i<4; i++) {
  Person person = persons[i];
  String lastName = person.getLastName();
  String firstName = person.getFirstName();
  int age = person.getAge();
  System.out.println(lastName + ", " + firstName + ". Age:" + age);
}

Arrays.sort(persons);

System.out.println();
System.out.println("Sorted by age");

for (int i=0; i<4; i++) {
  Person person = persons[i];
  String lastName = person.getLastName();
  String firstName = person.getFirstName();
  int age = person.getAge();
  System.out.println(lastName + ", " + firstName + ". Age:" + age);
}
}
}
```


Class Testing

- Output program

```
Natural Order
Goodyear, Elvis. Age:56
Clark, Stanley. Age:8
Graff, Jane. Age:16
Goodyear, Nancy. Age:69

Sorted by age
Clark, Stanley. Age:8
Graff, Jane. Age:16
Goodyear, Elvis. Age:56
Goodyear, Nancy. Age:69
BUILD SUCCESSFUL (total time: 2 seconds)
```

Menggunakan class Comparator

- Dengan mengimplementasikan interface Comparable kita hanya bisa menentukan satu cara saja untuk membandingkan object-object dari class Person, untuk contoh sebelumnya, yang kita bandingkan berdasarkan umur.
- Bagaimana jika object-object dari class Person diurutkan berdasarkan umur, nama awal dan nama akhir? Berarti object-object tersebut dibandingkan berdasarkan umur, nama awal dan nama akhir.
- Kita masih memerlukan dua cara lagi untuk membandingkan object-object dari class Person. Kita perlu dua 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

Menggunakan class Comparator

- Membuat comparator berdasarkan firstname. Jika firstName antara object 1 dengan object 2 tidak sama, maka yang dibandingkan adalah firstName, tetapi jika sama maka yang dibandingkan adalah lastName

```
package comparable.ex02;

import java.util.Comparator;


public class FirstNameComparator implements Comparator {
  public int compare(Object person, Object anotherPerson) {
    String lastName1 = ((Person) person).getLastName().toUpperCase();
    String firstName1 = ((Person) person).getFirstName().toUpperCase();
    String lastName2 = ((Person) anotherPerson).getLastName().toUpperCase();
    String firstName2 = ((Person) anotherPerson).getFirstName().toUpperCase();
    if (!(firstName1.equals(firstName2)))
      return firstName1.compareTo(firstName2);
    else
      return lastName1.compareTo(lastName2);
  }
}
```

Menggunakan class Comparator

- Membuat comparator berdasarkan lastname. Jika lastName antara object 1 dengan object 2 tidak sama, maka yang dibandingkan adalah lastName, tetapi jika sama maka yang dibandingkan adalah firstName

```
package comparable.ex02;

import java.util.Comparator;

public class LastNameComparator implements Comparator {
  public int compare(Object person, Object anotherPerson) {
    String lastName1 = ((Person) person).getLastName().toUpperCase();
    String firstName1 = ((Person) person).getFirstName().toUpperCase();
    String lastName2 = ((Person) anotherPerson).getLastName().toUpperCase();
    String firstName2 = ((Person) anotherPerson).getFirstName().toUpperCase();

    if (!(lastName1.equals(lastName2)))
      return lastName1.compareTo(lastName2);
    else
      return firstName1.compareTo(firstName2);
  }
}
```



```
package comparable.ex02;
import java.util.Arrays;

public class Testing {
     public static void main(String[] args) {
         Person[] persons = new Person[4];
         persons[0] = new Person();
         persons[0].setFirstName("Elvis");
         persons[0].setLastName("Goodyear");
         persons[0].setAge(56);

         persons[1] = new Person();
         persons[1].setFirstName("Stanley");
         persons[1].setLastName("Clark");
         persons[1].setAge(8);

         persons[2] = new Person();
         persons[2].setFirstName("Jane");
         persons[2].setLastName("Graff");
         persons[2].setAge(16);

         persons[3] = new Person();
         persons[3].setFirstName("Nancy");
         persons[3].setLastName("Goodyear");
         persons[3].setAge(69);
```

```
System.out.println("Natural Order");
for (int i = 0; i < 4; i++) {
    Person person = persons[i];
    String lastName = person.getLastName();
    String firstName = person.getFirstName();
    int age = person.getAge();
    System.out.println(lastName + ", " + firstName + ". Age:" + age);
}

Arrays.sort(persons, new LastNameComparator());
System.out.println();
System.out.println("Sorted by last name");

for (int i = 0; i < 4; i++) {
    Person person = persons[i];
    String lastName = person.getLastName();
    String firstName = person.getFirstName();
    int age = person.getAge();
    System.out.println(lastName + ", " + firstName + ". Age:" + age);
}
```



```
        Arrays.sort(persons, new FirstNameComparator());
        System.out.println();
        System.out.println("Sorted by first name");

        for (int i = 0; i < 4; i++) {
            Person person = persons[i];
            String lastName = person.getLastName();
            String firstName = person.getFirstName();
            int age = person.getAge();
            System.out.println(lastName + ", " + firstName + ". Age:" + age);
        }

        Arrays.sort(persons);
        System.out.println();
        System.out.println("Sorted by age");

        for (int i = 0; i < 4; i++) {
            Person person = persons[i];
            String lastName = person.getLastName();
            String firstName = person.getFirstName();
            int age = person.getAge();
            System.out.println(lastName + ", " + firstName + ". Age:" + age);
        }
    }
}
```

Menggunakan class Comparator

```
Natural Order
Goodyear, Elvis. Age:56
Clark, Stanley. Age:8
Graff, Jane. Age:16
Goodyear, Nancy. Age:69

Sorted by last name
Clark, Stanley. Age:8
Goodyear, Elvis. Age:56
Goodyear, Nancy. Age:69
Graff, Jane. Age:16

Sorted by first name
Goodyear, Elvis. Age:56
Graff, Jane. Age:16
Goodyear, Nancy. Age:69
Clark, Stanley. Age:8

Sorted by age
Clark, Stanley. Age:8
Graff, Jane. Age:16
Goodyear, Elvis. Age:56
Goodyear, Nancy. Age:69
```


Menggabungkan Comparator pada class Comparable

- Contoh sebelumnya masih memerlukan beberapa class. Bagaimana kita menggabungkan comparator dalam class comparable ?

```
package comparable.ex03;
import java.util.Comparator;
public class Person implements Comparable {
  private String firstName;
  private String lastName;
  private int age;

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  public int getAge() {
    return age;
  }

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


Menggabungkan Comparator pada class Comparable

```
public int compareTo(Object anotherPerson) throws ClassCastException {
  if (!(anotherPerson instanceof Person))
    throw new ClassCastException("A Person object expected.");
  int anotherPersonAge = ((Person) anotherPerson).getAge();
  return this.age - anotherPersonAge;
}

public static Comparator LastNameComparator = new Comparator() {
  public int compare(Object person, Object anotherPerson) {
    String lastName1 = ((Person) person).getLastName().toUpperCase();
    String firstName1 = ((Person) person).getFirstName().toUpperCase();
    String lastName2 = ((Person) anotherPerson).getLastName().toUpperCase();
    String firstName2 = ((Person) anotherPerson).getFirstName().toUpperCase();

    if (!(lastName1.equals(lastName2)))
      return lastName1.compareTo(lastName2);
    else
      return firstName1.compareTo(firstName2);
  }
};
```

Menggabungkan Comparator pada class Comparable

```
public static Comparator FirstNameComparator = new Comparator() {
  public int compare(Object person, Object anotherPerson) {
    String lastName1 = ((Person) person).getLastName().toUpperCase();
    String firstName1 = ((Person) person).getFirstName().toUpperCase();
    String lastName2 = ((Person) anotherPerson).getLastName().toUpperCase();
    String firstName2 = ((Person) anotherPerson).getFirstName().toUpperCase();

    if (!(firstName1.equals(firstName2)))
      return firstName1.compareTo(firstName2);
    else
      return lastName1.compareTo(lastName2);
  }
};
}
```


Menggabungkan Comparator pada class Comparable

- Setelah digabungkan untuk mengurutkan data berdasarkan lastname dengan cara

```
Arrays.sort(persons,
Person.LastNameComparator);
```

- untuk mengurutkan data berdasarkan firstname dengan cara:

- `Arrays.sort(persons, Person.FirstNameComparator);`