AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

19CSE204
Object Oriented Paradigm
2-0-3-3

Java Collections

- Sorting
- Comparable
- Comparator

Java Comparable interface

Java Comparable interface

- Java Comparable interface is used to order the objects of the user-defined class. This interface is found in java.lang package and contains only one method named compareTo(Object). It provides a single sorting sequence only, i.e., you can sort the elements on the basis of single data member only.
- **compareTo (Object obj) method**
  - **public int compareTo(Object obj):** It is used to compare the current object with the specified object. It returns
  - positive integer, if the current object is greater than the specified object.
  - negative integer, if the current object is less than the specified object.
  - zero, if the current object is equal to the specified object.

For a list of custom objects Collections.sort(List list) will give a Compile Time Error because there is no natural ordering defined for the corresponding object. Hence we have to implement comparable interface and define sorting procedure in CompareTo(Obj) method

Java Comparable Example

```java
package SortCompare;
import java.util.*;
class Student implements Comparable<Student>{
int rollno;
String name;
int age;
Student(int rollno,String name,int age){
this.rollno=rollno;
this.name=name;
this.age=age;
}
public int compareTo(Student st){
if(age==st.age)
return 0;
else if(age>st.age)
return 1;
else
return -1;
}
}
public class SortList {

    public static void main(String[] args) {
    ArrayList<Student> al=new ArrayList<Student>();
    al.add(new Student(101,"Vijay",23));
    al.add(new Student(106,"Ajay",27));
    al.add(new Student(105,"Jai",21));
    for(Student st:al){
        System.out.println(st.rollno+" "+st.name+" "+st.age);
        }
    Collections.sort(al);
    System.out.println("Aftersorting");
    for(Student st:al){
    System.out.println(st.rollno+" "+st.name+" "+st.age);
    }
```

Output

```
101 Vijay 23
106 Ajay 27
105 Jai 21
Aftersorting
105 Jai 21
101 Vijay 23
106 Ajay 27
```

Java Comparator interface

Sorting List elements

- We can sort the elements of:
  - String objects
  - Wrapper class objects
  - User-defined class objects
- Method of Collections class for sorting List elements
  - There are two overloaded Collections.sort() methods, which are:
    - **sort(List list):** Sorts the elements of the List in ascending order of their natural ordering.
    - **sort(List list, Comparator c):** Sorts the elements of the list according to the order induced by the comparator.

Note: String class and Wrapper classes implement the Comparable interface by default. So if you store the objects of string or wrapper classes in a list, set or map, it will be Comparable by default.

Methods of Java Comparator Interface

Method	Description
public int compare(Object obj1, Object obj2)	It compares the first object with the second object.
public boolean equals(Object obj)	It is used to compare the current object with the specified object.
public boolean equals(Object obj)	It is used to compare the current object with the specified object

Collections class provides static methods for sorting the elements of a collection. If collection elements are of Set or Map, we can use TreeSet or TreeMap. However, we cannot sort the elements of List. Collections class provides methods for sorting the elements of List type elements also.

Method of Collections class for sorting List elements

public void sort(List list, Comparator c): is used to sort the elements of List by the given Comparator.

Java Comparator Example (Non-generic Old Style)

- Let's see the example of sorting the elements of List on the basis of age and name. In this example, we have created 4 java classes:

1.Student.java

2.AgeComparator.java

3.NameComparator.java

4.Simple.java

Java Comparator Example (Non-generic Old Style)

```java
package comparatornongenerics;

public class Student {

    int rollno;
    String name;
    int age;
    Student(int rollno,String name,int age){
    this.rollno=rollno;
    this.name=name;
    this.age=age;
    }
    }
```

1.Student.java

```java
package comparatornongenerics;
import java.util.*;
public class AgeComparator implements Comparator{
    public int compare(Object o1,Object o2){
        Student s1=(Student)o1;
        Student s2=(Student)o2;

        if(s1.age==s2.age)
        return 0;
        else if(s1.age>s2.age)
        return 1;
        else
        return -1;
        }
        }
```

1.AgeComparator.java

```java
package comparatornongenerics;
import java.util.*;
public class NameComparator implements Comparator{
    public int compare(Object o1,Object o2){
        Student s1=(Student)o1;
        Student s2=(Student)o2;

        return s1.name.compareTo(s2.name);
        }
        }
```

1.NameComparator.java

```java
package comparatornongenerics;
import java.util.*;
import java.io.*;
public class Simple {

    public static void main(String[] args) {
        ArrayList al=new ArrayList();
        al.add(new Student(101,"Vijay",23));
        al.add(new Student(106,"Ajay",27));
        al.add(new Student(105,"Jai",21));

        System.out.println("Sorting by Name");

        Collections.sort(al,new NameComparator());
        Iterator itr=al.iterator();
        while(itr.hasNext()){
        Student st=(Student)itr.next();
        System.out.println(st.rollno+" "+st.name+" "+st.age);
        }

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

        Collections.sort(al,new AgeComparator());
        Iterator itr2=al.iterator();
        while(itr2.hasNext()){
        Student st=(Student)itr2.next();
        System.out.println(st.rollno+" "+st.name+" "+st.age);
        }


        }
        }
```

1.Simple.java

```
Sorting by Name
106 Ajay 27
105 Jai 21
101 Vijay 23
Sorting by age
105 Jai 21
101 Vijay 23
106 Ajay 27
```

Java Comparator Example (generic Style)

```java
package comparator;

public class Student {
    int rollno;
    String name;
    int age;
    Student(int rollno,String name,int age){
    this.rollno=rollno;
    this.name=name;
    this.age=age;
}

}
```

1.Student.java

```java
package comparator;
import java.util.*;
public class AgeComparator implements Comparator<Student>{

    public int compare(Student s1,Student s2){
        if(s1.age==s2.age)
        return 0;
        else if(s1.age>s2.age)
        return 1;
        else
        return -1;
        }
        }
```

1.AgeComparator.java

```java
package comparator;
import java.util.*;
public class NameComparator implements Comparator<Student>{
    public int compare(Student s1,Student s2){
        return s1.name.compareTo(s2.name);
        }
}
```

1.NameComparator.java

```java
package comparator;
import java.util.*;
import java.io.*;
public class Simple {

    public static void main(String[] args) {
        ArrayList<Student> al=new ArrayList<Student>();
        al.add(new Student(101,"Vijay",23));
        al.add(new Student(106,"Ajay",27));
        al.add(new Student(105,"Jai",21));

        System.out.println("Sorting by Name");

        Collections.sort(al,new NameComparator());
        for(Student st: al){
        System.out.println(st.rollno+" "+st.name+" "+st.age);
        }

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

        Collections.sort(al,new AgeComparator());
        for(Student st: al){
        System.out.println(st.rollno+" "+st.name+" "+st.age);
        }
        }
        }
```

1.Simple.java

```
Output
Sorting by Name
106 Ajay 27
105 Jai 21
101 Vijay 23
Sorting by age
105 Jai 21
101 Vijay 23
106 Ajay 27
```

Namah Shivaya