UTN - FRP

# Programación III – Java Collection Framework: sorting

El ordenamiento de una colección de objetos es un tema crucial en computación, y en Java está soportado especialmente en el Collection Framework. El lenguaje nos permite ordenar los siguientes tipos de colecciones:

1. Objetos Strings
2. Objetos Wrapper class
3. Objetos de clases definidas por el usuario

La clase **Collections** proporciona métodos estáticos para ordenar los elementos de una colección. Si los elementos de la colección son de tipo **Set**, podemos usar **TreeSet**. Sin embargo, no podemos ordenar los elementos de **List** directamente. La clase **Collections** proporciona métodos para ordenar los elementos de tipo Lista. Por simplicidad obviaremos las colecciones del tipo 1 y 2, y nos enfocaremos en las colecciones del tipo 3.

Ejemplo: ordenamiento de una lista de Student

// Java program to demonstrate working of Comparator

// interface and Collections.sort() to sort according

// to user defined criteria.

import java.util.\*;

import java.lang.\*;

import java.io.\*;

// A class to represent a student.

class Student

{

int rollno;

String name, address;

// Constructor

public Student(int rollno, String name,

String address)

{

this.rollno = rollno;

this.name = name;

this.address = address;

}

public int getRollno() {

return this.rollno;

}

// Used to print student details in main()

public String toString()

{

return this.rollno + " " + this.name +

" " + this.address;

}

}

class Sortbyroll implements Comparator<Student>

{

// Used for sorting in ascending order of

// roll number

public int compare(Student a, Student b)

{

return a.rollno - b.rollno;

}

}

// Driver class

class Main

{

public static void main (String[] args)

{

ArrayList<Student> ar = new ArrayList<Student>();

ar.add(new Student(111, "bbbb", "london"));

ar.add(new Student(131, "aaaa", "nyc"));

ar.add(new Student(121, "cccc", "jaipur"));

System.out.println("Unsorted");

for (int i=0; i<ar.size(); i++)

System.out.println(ar.get(i));

**Collections**.**sort**(ar, new Sortbyroll());

System.out.println("\nSorted by rollno");

for (int i=0; i<ar.size(); i++)

System.out.println(ar.get(i));

}

}

En este caso, **Collections**.sort funciona con cualquier clase que implementa la interfaz List. La clase **Sortbyroll** implementa la interfaz **Comparator** y es la que proporciona el método de ordenamiento por el atributo **rollno**.

A su vez, podemos eliminar la clase **Sortbyroll** y acortar nuestro código dado que la interfaz **Comparator** nos permite usar algunos métodos con el prefijo *comparing\** que nos retorna un objeto comparador con el criterio que le pasamos como argumento. Para este propósito se utiliza la referencia de funciones y/o funciones lambda.

Funciones por referencia:

**Collections**.**sort**(colección, **Comparator**.*comparingInt*(Student::getRollno));

Funciones lambda:

**Collections**.**sort**(colección, **Comparator**.comparingInt((Student a)-> {

return a.getRollno();

}));

Referencia:

<https://www.geeksforgeeks.org/comparator-interface-java/>

<https://www.geeksforgeeks.org/collections-sort-java-examples/>

<https://www.javatpoint.com/Sorting-in-collection-framework>