**Comparable and Comparator**

**Comparable and Comparator** in Java are very useful for sorting collection of objects. Java provides some inbuilt methods to sort primitive types array or Wrapper classes array or list.

**JavaSorting.java:**

package com.sort;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class JavaSorting {

public static void main(String[] args) {

//sort primitives array like int array

int[] intArr = {5,9,1,10};

Arrays.sort(intArr);

System.out.println(Arrays.toString(intArr));

//sorting String array

String[] strArr = {"A", "C", "B", "Z", "E"};

Arrays.sort(strArr);

System.out.println(Arrays.toString(strArr));

//sorting list of objects of Wrapper classes

List<String> strList = new ArrayList<String>();

strList.add("A");

strList.add("C");

strList.add("B");

strList.add("Z");

strList.add("E");

Collections.sort(strList);

for(String str: strList) System.out.print(" "+str);

}

}

Now let’s try to sort an array of objects

**Employee.java:**

package com.sort;

public class Employee {

private int id;

private String name;

private int age;

private long salary;

public int getId() {

return id;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public long getSalary() {

return salary;

}

public Employee(int id, String name, int age, int salary) {

this.id = id;

this.name = name;

this.age = age;

this.salary = salary;

}

@Override

//this is overridden to print the user-friendly information about the Employee

public String toString() {

return "[id=" + this.id + ", name=" + this.name + ", age=" + this.age + ", salary=" +

this.salary + "]";

}

}

Here is the code I used to sort the array of Employee objects.

//sorting object array

Employee[] empArr = new Employee[4];

empArr[0] = new Employee(15, " Bikash ", 20, 8000);

empArr[1] = new Employee(20, " Abshishek ", 25, 10000);

empArr[2] = new Employee(6, "Ganesh", 30, 15000);

empArr[3] = new Employee(2, "Debopriyo", 12, 40000);

//sorting employees array using Comparable interface implementation

Arrays.sort(empArr);

System.out.println("Default Sorting of Employees list:\n"+Arrays.toString(empArr));

Throwing Error.

Java provides **Comparable** interface which should be implemented by any custom class if we want to use Arrays or Collections sorting methods. The Comparable interface has **compareTo(T obj)** method which is used by sorting methods,

After implementing Comparable [interface](https://www.journaldev.com/1601/interface-in-java) in Employee class, here is the resulting Employee class.

**Employee.java:**

package com.sort;

import java.util.Comparator;

public class Employee implements Comparable<Employee> {

private int id;

private String name;

private int age;

private long salary;

public int getId() {

return id;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public long getSalary() {

return salary;

}

public Employee(int id, String name, int age, int salary) {

this.id = id;

this.name = name;

this.age = age;

this.salary = salary;

}

@Override

public int compareTo(Employee emp) {

//let's sort the employee based on an id in ascending order

//returns a negative integer, zero, or a positive integer as this employee id

//is less than, equal to, or greater than the specified object.

return (this.id - emp.id);

}

@Override

//this is required to print the user-friendly information about the Employee

public String toString() {

return "[id=" + this.id + ", name=" + this.name + ", age=" + this.age + ", salary=" +

this.salary + "]";

}

}

Now when we execute the above snippet for Arrays sorting of Employees and print it, here is the output.

Default Sorting of Employees list:

Output is based on ID.

As you can see that Employees array is sorted by id in ascending order

But, in most real-life scenarios, we want sorting based on different parameters. For example, as a CEO, I would like to sort the employees based on Salary, an HR would like to sort them based on the age. This is the situation where we need to use **Java Comparator** interface because *Comparable.compareTo(Object o)* method implementation can provide default sorting and we can’t change it dynamically. Whereas with Comparator, we can define multiple methods with different ways of sorting and then chose the sorting method based on our requirements.

**So if we want to sort the collections based on multiple criteria/fields then we have to use Comparator only**.

**Employee.java**:

package com;

public class Employee {

int id;

String name,address;

public Employee(int id, String name, String address) {

super();

this.id = id;

this.name = name;

this.address = address;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", address=" + address

+ "]";

}

}

**EmployeeByID.java**:

package com;

import java.util.Comparator;

import java.util.function.Function;

import java.util.function.ToDoubleFunction;

import java.util.function.ToIntFunction;

import java.util.function.ToLongFunction;

public class EmployeeByID implements Comparator{

public int compare(Object arg0, Object arg1) {

Employee e1=(Employee)arg0;

Employee e2=(Employee)arg1;

return e1.id-e2.id;

}

public static Comparator comparing(Function arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparing(Function arg0, Comparator arg1) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparingDouble(ToDoubleFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparingInt(ToIntFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparingLong(ToLongFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator naturalOrder() {

// TODO Auto-generated method stub

return null;

}

public static Comparator nullsFirst(Comparator arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator nullsLast(Comparator arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator reverseOrder() {

// TODO Auto-generated method stub

return null;

}

public Comparator reversed() {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparing(Comparator arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparing(Function arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparing(Function arg0, Comparator arg1) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparingDouble(ToDoubleFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparingInt(ToIntFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparingLong(ToLongFunction arg0) {

// TODO Auto-generated method stub

return null;

}

}

**EmployeeByName.java**

package com;

import java.util.Comparator;

import java.util.function.Function;

import java.util.function.ToDoubleFunction;

import java.util.function.ToIntFunction;

import java.util.function.ToLongFunction;

public class EmployeeByName implements Comparator{

public int compare(Object arg0, Object arg1) {

Employee e1=(Employee)arg0;

Employee e2=(Employee)arg1;

return e1.name.compareTo(e2.name);

}

public static Comparator comparing(Function arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparing(Function arg0, Comparator arg1) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparingDouble(ToDoubleFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparingInt(ToIntFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator comparingLong(ToLongFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator naturalOrder() {

// TODO Auto-generated method stub

return null;

}

public static Comparator nullsFirst(Comparator arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator nullsLast(Comparator arg0) {

// TODO Auto-generated method stub

return null;

}

public static Comparator reverseOrder() {

// TODO Auto-generated method stub

return null;

}

public Comparator reversed() {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparing(Comparator arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparing(Function arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparing(Function arg0, Comparator arg1) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparingDouble(ToDoubleFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparingInt(ToIntFunction arg0) {

// TODO Auto-generated method stub

return null;

}

public Comparator thenComparingLong(ToLongFunction arg0) {

// TODO Auto-generated method stub

return null;

}

}

**EmpMain.java:**

package com;

import java.util.\*;

public class EmpMain {

public static void main(String[] args) {

Employee e1=new Employee(1,"Shanka","Tollygunge");

Employee e2=new Employee(2,"Bapan","Selimpur");

Employee e3=new Employee(3,"Asim","Jadavpur");

Employee e4=new Employee(4,"Pritam","Dumdum");

ArrayList<Employee> list=new ArrayList<Employee>();

list.add(e1);

list.add(e4);

list.add(e2);

list.add(e3);

System.out.println("Before sorting the List based on Id::"+list);

Collections.sort(list,new EmployeeByID());

System.out.println("After sorting the List based on Id::"+list);

list.clear();

list.add(e1);

list.add(e4);

list.add(e2);

list.add(e3);

System.out.println("Before sorting the List based on Name::"+list);

Collections.sort(list,new EmployeeByName());

System.out.println("After sorting the List based on Name::"+list);

}

}

**OutPut:**

**Before sorting the List based on Id**::[Employee [id=1, name=Shanka, address=Tollygunge], Employee [id=4, name=Pritam, address=Dumdum], Employee [id=2, name=Bapan, address=Selimpur], Employee [id=3, name=Asim, address=Jadavpur]]

**After sorting the List based on Id**::[Employee [id=1, name=Shanka, address=Tollygunge], Employee [id=2, name=Bapan, address=Selimpur], Employee [id=3, name=Asim, address=Jadavpur], Employee [id=4, name=Pritam, address=Dumdum]]

**Before sorting the List based on Name**::[Employee [id=1, name=Shanka, address=Tollygunge], Employee [id=4, name=Pritam, address=Dumdum], Employee [id=2, name=Bapan, address=Selimpur], Employee [id=3, name=Asim, address=Jadavpur]]

**After sorting the List based on Name**::[Employee [id=3, name=Asim, address=Jadavpur], Employee [id=2, name=Bapan, address=Selimpur], Employee [id=4, name=Pritam, address=Dumdum], Employee [id=1, name=Shanka, address=Tollygunge]]