# Java & JEE Training

## Day 16 – Collections – Sorting, Comparing

MindsMapped Consulting

# Agenda

- Recap of Arrays, ArrayLists
- Basically, there can be 2 operations that you would want to perform on Arrays/ArrayLists.. (and maybe, other collections)
- Search: Override equals() and hashCode().
- Sort: provide comparison logic – Two ways
  - Comparable interface
  - Comparator interface

# Comparable vs Comparator Interfaces

| Comparable | Comparator |
|---|---|
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class** i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method** to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by **Collections.sort(List)** method. | We can sort the list elements of Comparator type by **Collections.sort(List,Comparator)** method. |

# Comparable Interface Example

```
public class Person implements Comparable {
    private int person_id;
    private String name;

    /**
     * Compare current person with specified person
     * return zero if person_id for both person is same
     * return negative if current person_id is less than specified one
     * return positive if specified person_id is greater than specified one
     */
    @Override
    public int compareTo(Object o) {
        Person p = (Person) o;
        return this.person_id - o.person_id ;
    }
    ….
}
```

# Comparator Interface

```
/**
 * Comparator implementation which sorts Person objects on person_id field
 */
public class SortByPerson_ID implements Comparator{

    public int compare(Object o1, Object o2) {
        Person p1 = (Person) o;
        Person p2 = (Person) o;
        return p1.getPersonId() - p2.getPersonId();
    }
}
```

# Comparing in Java – Strings and Dates

- Strings are immutable
- String implements Comparable interface
- Call String1.compareTo(String2)

- Date also implements Comparable.
- Date1.compareTo(Date2)

# Where is the Comparator or Comparable interfaces used?

| Constructor | Description |
|---|---|
| TreeSet() | It is used to construct an empty tree set that will be sorted in an ascending order according to the natural order of the tree set. |
| TreeSet(Collection c) | It is used to build a new tree set that contains the elements of the collection c. |
| TreeSet(Comparator comp) | It is used to construct an empty tree set that will be sorted according to given comparator. |
| TreeSet(SortedSet ss) | It is used to build a TreeSet that contains the elements of the given SortedSet. |

# Exercise...

1. Create an employee class with fields – id, name, age, salary.
2. Provide default comparison logic for id using java.lang.Comparable interface.
3. Provide the following comparison logic using java.util.Comparator interface
   - Name
   - Age
   - Salary
   - **Name and then Age.**
4. Test the above using
   - Array of Employees and using Arrays.sort()
   - ArrayList of Employees and using Collections.sort()