**Comparable و Comparator در جاوا:**

زبان جاوا برای مرتب کردن اشیاء بر اساس دیتای آن اشیا دو اینترفیس به نام های Comparable و Comparator ارائه کرده است.

* **Comparable :**

یک شئ از این جنس توانایی قیاس خود را با شئ دیگر داراست. کلاس مربوط به این شئ بایستی اینترفیس  **java.lang.Comparable را اینپلیمنت کند.** یک کلاس فیلم را با state هایی نظیر نام ، سال و میزان پسند در نظر بگیرید. حال این را در نظر بگیرید که میخواهیم لیستی از اشیاء ساخته شده از این کلاس را برا اساس سال مرتب کنیم. ما میتوانیم این اینترفیس را به کلاس مربوط اینپلیمنت کنیم. سپس متد compareTo() وابسته به آن را بازنویسی(override) نماییم.

Movies after sorting :

Star Wars 8.7 1977

Empire Strikes Back 8.8 1980

Return of the Jedi 8.4 1983

Force Awakens 8.3 2015

//import java.io.\*;

import java.util.\*;

// A class 'Movie' that implements Comparable

class Movie implements Comparable<Movie>

{

private double rating;

private String name;

private int year;

// Used to sort movies by year

public int compareTo(Movie m)

{

return this.year - m.year;

}

// Constructor

public Movie(String nm, double rt, int yr)

{

this.name = nm;

this.rating = rt;

this.year = yr;

}

// Getter methods for accessing private data

public double getRating() { return rating; }

public String getName() { return name; }

public int getYear() { return year; }

}

// Driver class

class Main

{

public static void main(String[] args)

{

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

list.add(new Movie("Force Awakens", 8.3, 2015));

list.add(new Movie("Star Wars", 8.7, 1977));

list.add(new Movie("Empire Strikes Back", 8.8, 1980));

list.add(new Movie("Return of the Jedi", 8.4, 1983));

Collections.sort(list);

System.out.println("Movies after sorting : ");

for (Movie movie: list)

{

System.out.println(movie.getName() + " " + movie.getRating() + " " + movie.getYear());

}

}

}

* **Comparator:**

بر خلاف اینترفیس بالا، این اینترفیس یک عنصر خارجی برای قیاس اشیاء مدّنظر ماست. برای این منظور ما کلاس های جداگانه ای نیاز داریم تا از این اینترفیس استفاده کنیم.

1. باید کلاسی بسازیم تا از comparator اینپلیمنت سازی کند.
2. از روی کلاس ساخته شده شئ خاصی میسازیم.
3. متد بازنویسی شده sort را که برای اشیاء قبلی استفاده کرده ایم فراخوانی میکنیم.

بر اساس مثال قبلی به کد زیر دقت کنید:

// Driver class

class Main

{

public static void main(String[] args)

{

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

list.add(new Movie("Force Awakens", 8.3, 2015));

list.add(new Movie("Star Wars", 8.7, 1977));

list.add(new Movie("Empire Strikes Back", 8.8, 1980));

list.add(new Movie("Return of the Jedi", 8.4, 1983));

// Sort by rating : (1) Create an object of ratingCompare

// (2) Call Collections.sort

// (3) Print Sorted list

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

RatingCompare ratingCompare = new RatingCompare();

Collections.sort(list, ratingCompare);

for (Movie movie: list)

System.out.println(movie.getRating() + " " +

movie.getName() + " " +

movie.getYear());

// Call overloaded sort method with RatingCompare

// (Same three steps as above)

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

NameCompare nameCompare = new NameCompare();

Collections.sort(list, nameCompare);

for (Movie movie: list)

System.out.println(movie.getName() + " " +

movie.getRating() + " " +

movie.getYear());

// Uses Comparable to sort by year

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

Collections.sort(list);

for (Movie movie: list)

System.out.println(movie.getYear() + " " +

movie.getRating() + " " +

movie.getName()+" ");

}

}

//A Java program to demonstrate Comparator interface

import java.io.\*;

import java.util.\*;

// A class 'Movie' that implements Comparable

class Movie implements Comparable<Movie>

{

private double rating;

private String name;

private int year;

// Used to sort movies by year

public int compareTo(Movie m)

{

return this.year - m.year;

}

// Constructor

public Movie(String nm, double rt, int yr)

{

this.name = nm;

this.rating = rt;

this.year = yr;

}

// Getter methods for accessing private data

public double getRating() { return rating; }

public String getName() { return name; }

public int getYear() { return year; }

}

// Class to compare Movies by ratings

class RatingCompare implements Comparator<Movie>

{

public int compare(Movie m1, Movie m2)

{

if (m1.getRating() < m2.getRating()) return -1;

if (m1.getRating() > m2.getRating()) return 1;

else return 0;

}

}

// Class to compare Movies by name

class NameCompare implements Comparator<Movie>

{

public int compare(Movie m1, Movie m2)

{

return m1.getName().compareTo(m2.getName());

}

}

Output :

Sorted by rating

8.3 Force Awakens 2015

8.4 Return of the Jedi 1983

8.7 Star Wars 1977

8.8 Empire Strikes Back 1980

Sorted by name

Empire Strikes Back 8.8 1980

Force Awakens 8.3 2015

Return of the Jedi 8.4 1983

Star Wars 8.7 1977

Sorted by year

1977 8.7 Star Wars

1980 8.8 Empire Strikes Back

1983 8.4 Return of the Jedi

2015 8.3 Force Awakens