```
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());
        }
    }
}
```

1. Consider a Movie class that has members like, rating, name, year. Functions of the class:
public double getRating()
public String getName()
public int getYear()

Suppose we wish to sort a list of Movies based on year of release.
We can implement the Comparable interface with the Movie class, and we override the method compareTo() of Comparable interface.

```
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() + " ");
    }
}
```

2. Now, suppose we want to sort movies by their rating and names as well. When we make a collection element comparable(by having it implement Comparable), we get only one chance to implement the compareTo() method. The solution is using Comparator.

To compare movies by Rating, we need to do 3 things:
i. Create a class that implements Comparator (and thus the compare() method that does the work previously done by compareTo()).
class RatingCompare
class NameCompare
ii. Make an instance of the Comparator class.
iii. Call the overloaded sort() method, giving it both the list and the instance of the class that implements Comparator.