**Let’s create a bug | collections.sort() | Selenium Express**

**Sorting your Customize Object**

**package** sorting.seleniumexpress;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** StringSortRemoveDuplicateCode {

**public** **static** **void** main(String[] args) {

ArrayList <String> musicList = **new** ArrayList<String>();

// this is not sorted

musicList.add("zara zara");

musicList.add("besabriyaan");

musicList.add("kaise yua");

musicList.add("aeisa desh hai mera");

musicList.add("koi lauta de woh pyare din");

MyUtil.*iterateList*(musicList);

System.***out***.println();

//---------------------

Collections.*sort*(musicList);

System.***out***.println("After Sorting : --- ");

MyUtil.*iterateList*(musicList);

}}

------------------------x-----------------------------------x------------------------

**Example –**

Here we have a class called ‘StringSortRemoveDuplicateCode’ and we are adding some music to our Array List and We are just giving the Title here ie these are my music title, like zara zara, besabriyaan.

Let’s say My client come back and he basically told that “I want to add store some more information about my song and I just don’t want to store the title and I also want to store who is the actual Artist for the particular song and also understood that when was that particular song released ”.

So Basically I want to store three different information about my song – who is the Artist, Release year and the Title of the song-

So I have to create my custom class 10.

1. Create a class [ Song] and Package [songAPI]
2. In the song class I will be storing three different things
3. Setter and Getter method to access this variable outside of my class. So I can go to source to generate setter and getter from my class
4. Go back to in a Project Explorer > right click > create a class with main () and the name like MP4 to create a Play List.
5. Create a Array List in MP4 class and this list will be storing all my songs . and the package name -Example.Sorting-[Separate package ]

ArrayList <song>musicList = **new** ArrayList <> ();

So I have to import this particular song class here as well which is there in my APISong package .

1. Create a song objects AND after that I will add all those objects and then I am getting an error , because here I don’t have this particular Constructor in my Song class.
2. Create a Constructor in my Song class.
3. I can add all these songs to my music class
4. Now use the add() of Array List lists , LIKE musicList
5. Iterate over the music list and I want to see whether all these songs are present in my List or not.
6. I already created a method there is my MyUtil class and it can iterate any kind of List and I can pass on this musicList to this method and hopefully this iterate list will work fine .

This particular method can able to handle any kind of List and can iterate over it .

So this method can take any list and can iterate and can iterate over this . and in this method[ in the MyUtil class] ‘T ‘will become song and it will pass a list of song and it will iterate and print it to my console.

1. So Basically I am getting the object reference instead of all these information.
2. But I want Whenever I will print my song object you should not give the reference of the object
3. I want to give the content of the object .So if you see here in the MyUtil .java. It is basically printing the temp [object reference of Common method ] . here T will be song and and the reference of T = temp would be song.

So The Song reference whenever we are printing it , it is basically printing the object reference.

1. Now we want to print the information about the song class. So go the song class and In the song class I am going to override the to string method [toString() ] . That’s why go to source to generate out two String and I want all these information to be printed. \
2. So My Id will generate a toString() for me.and then we can see this is the toStrint() and Basically it is printing the title, year , artists .

Note--

So All my fields and properties which I have created –

**private** String title;

**private** String artist;

**private** Integer year;

1. Collections.*sort*(musicList);

here I see that an error method is coming –

So Whenever I am creating an Array List of song which is my custom object --[ArrayList <**Song>** musicList = **new** ArrayList <> ();] and which is custom class. This song is my custom class.So this particular list whenever I am passing it to sort . I am getting an issue, it’s not able to compile and even if I am going to run then I will get this Error message on console—

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

The method sort(List<T>) in the type Collections is not applicable for the arguments (ArrayList<Song>)

at example.Soring.MP4.main(MP4.java:28)

**Q- Why This sort is giving us a compile time error ?**

**This music list is basically our ArrayList which contains a list of Song. So we have to implement Comparable Interface to our Song class and then we have to override and compareTo method in the Song class. Then there will be no error. And then syso to see after soring output but it is not even sorted and it’s not get sorted.**

**Sots method is not working here because of this things -**<T extends [**Comparable**](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)<? super T>>.

**Before I fix this particular things -** Collections.*sort*(musicList);

**The sort method is only excepting those kinds of list here which is extending to the Comparable Interface.**

[**sort**](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html#sort-java.util.List-)**(**[**List**](https://docs.oracle.com/javase/8/docs/api/java/util/List.html)**<T> list)-**

1. The sort() takes the List of any type that type or that T should extend to comparable.

That means Whatever the type that you are going to pass, it’s not a problem .

I can pass any list of object , it may be a list of Integer, or it may be a list of String or a list of song but if you are passing it make sure that particular class should implement an Interface called Comparable Interface .

Basically this comparable class is a Interface but I am written here [Java people will write here ] T extends comparable.

1. **Extends can be used for classes, it can also use for the Interface in generic not Everywhere.**
2. **So <T extends comparable> that means I can sort you any kind of list but whatever the type of the list you are sending make sure that particular class should extends or should implements an Interface called Comparable Interface.**
3. **In Eclipse previously we have sorted Integer . that why we did Collections.sort and we passed a list of Integer🡪(IntegerList) -- Collections.sort(integerList).**

**So If you go to Integer class and If I do ctrl + click her**

ArrayList <Integer> integerList = **new** ArrayList<Integer> (); then I see I am inside my Integer class. Ans this Integer class extends to Number [This number whatever I don’t care] but it’s implements an Interface called comparable .

So as a sort() says there 🡪 [T extends [**Comparable**](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html) **]T should be exchanged to Comparable. I.e I can sort the list but that list whatever the object that you were sending me with that list , that type of object should extends to Comparable .**

**So the Integer class implements an Interface called Comparable.**

**\*\* Similarly if I go to the String we have sorted String –Collections.sort(musicList);**

**So If I go to String class [Ctrl + click on the String]**

ArrayList <String> musicList = **new** ArrayList<String>();

So here is also we see that String class implements an Interface called Comparable.

So Now to MP4 class to solve the error Which I got.

In the MP4 class We are passing a list called musicList. If I will do command click on the musicList , here musicList is a type of Song. And this Song class is not implementing the Comparable Interface.

So I can just give a special implements and Comparable Interface.and I can give the type <> song type. Then I can see an Error . if I click there this is unimplemented methods. So this Comparable interface does have one method that method name is compareTo(). It will return int.

Now We can see that there is Error in the sort() in the MP4 class.

So the Problem was that We have not implemented the Comparable Interface to our Song class.

---------------

The Comparable Interface is inside the java.lang package .

Public final class Integer extends Number implements Comparable<Integer>{}

**Integer class extends number implements Comparable Interface but Implements an interface called Comparable.**

**[To understand this thing now I crate a Test class name -SortingTest class and in the SortingTest we have another class like Human class and Doctor class and Doctor is also a human . and another class here Animal class.Here Animal is not a human ,so we don’t need to extends this animal to the Human class and now we create another class here called class Dog and a Dog is a Animal .So Dog Extends Animal and Then I create an another class Bird ,So Bird is a not Animal .**

**And then go back to the Test class and inside the test class I want to create a method [public void printInfo(){}] to print the information something right here. So Basically this method is going to help you to print any kind of information about a Human , Animal, Doctor , Doctor and Bird or anything.**

**Now I am going to make this method a Generic method because it can accept any kind of , any type of objects and to make this method as a generic- and then syso and I can make this method static so that I can call this method Directly**

**Using Method name like printInfo(); and this particular method is accepting any kind of object. So I can pass a Human object like What is happing new Human.**

**Now I want to pass on A Doctor object . So I run this program then there is no problem , it’s compiled fine because this** *printInfo*(**); takes any kind of object. So it is also going to fine.**

**But My Client he wants to restrict that this particular method only can be used by the Human not anyone .**

**So right now this method is used by Everybody ----**

**public** **static** <T> **void** printInfo(T anyObj){

System.***out***.println("Hey you are a Human /Dog/Animal/Bird/");

}

**But the client wants this method to be used by this Human, I means anybody who is a Human. So Doctor is also a Human because it’s exchange to Human class.**

**So Basically human class and the subclass of human can actually use that particular method. So in this case I have to do only like that -**<T extends Human> .So this method will be like that

**public** **static** **<T extends Human> void** printInfo(T anyObj){

System.***out***.println("Hey you are a Human /Dog/Animal/Bird/");

}

So Now this T extends to human. So that you will get error message if you call any class other than human. Because T only extends only Human not any animal or …..

So I only can pass a human object and I can pass a Doctor object here. Because this particular syntax here T extend human. And I can pass the doctor object because Doctor is Exchanged to human.

**Question –How can the doctor can get all the human properties ? Obviously it’s needs to Exchange to human.**

**If you don’t know about it then go to google to the following— because we need to know a method called sort.**

Go to Java API 8 > Overview >Search for sort which is in the Collections class

And the Collections class present inside the java.util package> so first go to java package> java.util> Collections class> it has two sort method-

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | static <T extends [**Comparable**](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)<? super T>> void | [**sort**](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html#sort-java.util.List-)([**List**](https://docs.oracle.com/javase/8/docs/api/java/util/List.html)<T> list)  Sorts the specified list into ascending order,  according to the [**natural ordering**](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html) of its elements |  1. This sort method we are using in our collections class and this sort() is static. 2. There is an argument that is accepting any kind of List. This T tends any kind of type, it is going to accept. But if you see the ,method signature , if you don’t generate, if you never worked on generic. 3. Whatever the sort method does? Ans—Sorts the specified list into ascending order according to the natural ordering of its elements. 4. This is a void method. Actually it will return nothing 5. Extends Comparable   \*\* All these method in the collections class, i.e most of the methods are static.   |  |  |  | | --- | --- | --- | | static <T> void | [**sort**](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html#sort-java.util.List-java.util.Comparator-)([**List**](https://docs.oracle.com/javase/8/docs/api/java/util/List.html)<T> list, [**Comparator**](https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html)<? super T> c)  Sorts the specified list according to the order induced by the specified comparator. | | |  | | |

**Question –Why all these methods inside the collections class are static ?**

**Ans- Before we have created [Common]** iterateList **method in my MyUtil class that is also static because there’s the utility method.**

**public** **static** <T> **void** iterateList(ArrayList<T> anyList) {

**for** (T temp : anyList) {

System.***out***.println(temp);

}

**This method Sort() [and others method in the Collections class] are kind of utility method also. So they are static.**

**Q Why we are making all the Utility class as Static Method?**

**Ans-**

**package** APISong;

**public** **class** Song {

**private** String title;

**private** String artist;

**private** Integer year;

**public** Song(String title, String artist, **int** year) {

**this**.title=title;

**this**.artist =artist;

**this**.year = year;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getArtist() {

**return** artist;

}

**public** **void** setArtist(String artist) {

**this**.artist = artist;

}

**public** Integer getYear() {

**return** year;

}

**public** **void** setYear(Integer year) {

**this**.year = year;

}

@Override

**public** String toString() {

**return** "Song [title=" + title + ", artist=" + artist + ", year=" + year + "]";

}

}

# compareTo() hands-on | sorting custom objects in java

# Basically the Comparable is used for natural soring that means If you want to sort something in ascending order or the comparable is heavily is used for that particular thing. So this particular thing says that this Song class is right now ready but to compare two different Song objects and can sort them in a natural sorting order and that’s why we have to write our own logic in the @override compareTo().

So We know that CompareTo() returns an int .So there is a fundamental with this particular thing.

String class Implements Comparable Interface . If String class implemented the Comparable Interface then it also has the CompareTo() because this abstract method inside the comparable interface.

So String class if it is extend or if it is implementing the comparable interface then it would have a method here called CompareTo().

**In case of Integer ---**

[CompareTo() returns a positive integer or a negative Integer or a zero Integer. So Syso(“a”.complareTo(“x”)); , So in this case I have to think that “a” is coming before “x” or after “x”. So it is coming before x. So it is coming before and make sure it is going to give you a negative Integer.

So the conclusion that Negative Integer if the current objects is lesser then the specified object. So the current object means this “a” object. so a is lesser than x.

**In case of String---**

We are using CompareTo () over a string objects.So this compareTo() is inside the string class. So in the string class they have already retained the logic for all these things .

Just like that Integer class also implements the comparable interface. So I go to integer class also there is a compareTo() and the fundamental is Exactly same.

So String class has a compareTo() and Integer class has a compareTo() , if I want my Song class to be comparable , if I want my song objects to compare then my class also need to implement the comparable interface and I also need to give a valid logic to my CompareTo () method.

If(this,getYear()> cnotherSong.getYear()){

Return+1;

}

**//Q – how can I actually Implements that particular String?**

Task- Now don’t use Integer class compateTo().

Implement your own Logic .---

So I am writing my own logic depending on the Integer class compareTo().

If (this.getYear() > anotherSong.getYear()){ return +1;

}

// Current object is greater than the object we are comparing to obviously we need to return positive value. I can say positive 100 ,199 whatever value should be positive .

So

If (this.getYear() < anotherSong.getYear()){ return -1;}

}

return 0;

Otherwise if both these conditions doesn’t satisfy. Obviously both the objects ae equal ,so let me return zero.

return this . getTitle()

Current object title getTitle() and I want to compare this compareTo because this getTitle is

**@Override**

**public int compareTo(Song anotherSong) {**

**return this.getYear().compareTo(anotherSong.getYear());**

**}**

**Here “this” means current object. So my current object getYear dot compareTo**

**this.getYear()- It will give me an Integer. So In this Integer class there is already a CompareTo() and They have their own logic. So We have to just reuse their logic . I can use Integer class compareTo()and here I can pass another Integer.**

**Another Integer means this is one object 🡪|this.getYear()| and One object I will be getting through the compared to parentheses in the compared to arguments. So I can say public int compareTo(Song anotherSong){} and return this.getYear().compareTo(anotherSong.getYear());**

**So based on this compareTo() is going to return a positive number or negative number or a zero because this compared to method is already there inside the integer class. And an Integer class is already provided a valid implementation for the compareTo().**

**Or We can write like this to get the sorted value**

**@Override**

**public int compareTo(Song anotherSong) {**

**return this.getTitle().compareTo(anotherSong.getTitle());**

**}**

**Here Current object Title means this.getTitle and he have getter method –public String getTitle().So Title is string type and it will return String and In the String class we have a CompareTo().**

**getTitle will return String and In**

**So String class compareTo () will return a positive , negative number or zero based on whatever the thing we are passing right here, it will compare, it will do the things and it will return it to the sort().**

**So those are all for Ascending order.**

**Descending order---In this case we have to type only minus sign before this operator**

**@Override**

**public int compareTo(Song anotherSong) {**

**return -this.getTitle().compareTo(anotherSong.getTitle());**

**}**

**package** APISong;

**public** **class** Song **implements** Comparable <Song> {

**private** String title;

**private** String artist;

**private** Integer year;

**public** Song(String title, String artist, **int** year) {

**this**.title=title;

**this**.artist =artist;

**this**.year = year;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getArtist() {

**return** artist;

}

**public** **void** setArtist(String artist) {

**this**.artist = artist;

}

**public** Integer getYear() {

**return** year;

}

**public** **void** setYear(Integer year) {

**this**.year = year;

}

@Override

**public** String toString() {

**return** "Song [title=" + title + ", artist=" + artist + ", year=" + year + "]";

}

@Override

**public** **int** compareTo(Song anotherSong) {

//return this.getYear().compareTo(anotherSong.getYear());

//or

**return** **this**.getTitle().compareTo(anotherSong.getTitle()); // Ascending order

/\* or

\*

\* if(this.getYear() > anotherSong.getYear()); {

\* return +1; }

\*

\* if(this.getYear() < anotherSong.getYear()); {

\* return -1; }

\*

\* }

\* return 0 ;

\*/

}}

package example.Soring;

import java.util.ArrayList;

import java.util.Collections;

import APISong.Song;

import sorting.seleniumexpress.MyUtil;

public class MP4 {

public static void main(String[] args) {

Song song1 = new Song("take me to your heart","Michal Leans to Back",2014);

Song song2 = new Song("see you again","Wiz Khlif",2015);

Song song3 = new Song("love me like you do","Ellie Goulding",2013);

Song song4 = new Song("just a dream ", "Nelly", 2015);

Song song5 = new Song("as long as you live mew", "Nelly", 2010);

ArrayList <Song> musicList = new ArrayList <> ();

musicList.add(song1);

musicList.add(song2);

musicList.add(song3);

musicList.add(song4);

musicList.add(song5);

MyUtil.iterateList(musicList); // Before sorted

Collections.sort(musicList);

System.out.println("After Sorting---- ");

MyUtil.iterateList(musicList); //After sorted

}}

//Sort() will internally call the compareTo() of Song class.

**package** example.Soring;

**class** Human{

}

**class** Doctor **extends** Human{

}

**class** Animal {

}

**class** Dog **extends** Animal{

}

**class** Bird{

}

**public** **class** SortingTest {

/\*

\* public static <T> void printInfo(T anyObj){

\* System.out.println("Hey you are a Human /Dog/Animal/Bird/");

\* /

**public** **static** <T **extends** Human> **void** printInfo(T anyHumanObj){

System.***out***.println("Hey you are a Human ");

}

**public** **static** **void** main(String[] args) {

*printInfo*(**new** Human());

*printInfo*(**new** Doctor());

/\*

\* printInfo(new Dog()); printInfo(new Bird());

\*/

}

}