Java object sorting example (Comparable and Comparator)

By **mkyong** | July 7, 2010 | Last Updated : August 29, 2012

In this tutorial, it shows the use of **java.lang.Comparable** and **java.util.Comparator** to sort a Java object based on its property value.

1. Sort an Array

To sort an Array, use the **Arrays.sort()**.

```
        String[] fruits = new String[] {"Pineapple","Apple", "Orange", "Banana"};

        Arrays.sort(fruits);

        int i=0;
        for(String temp: fruits){
                System.out.println("fruits " + ++i + " : " + temp);
        }
```

Output

```
fruits 1 : Apple
fruits 2 : Banana
fruits 3 : Orange
fruits 4 : Pineapple
```

2. Sort an ArrayList

To sort an ArrayList, use the **Collections.sort()**.

```
        List<String> fruits = new ArrayList<String>();

        fruits.add("Pineapple");
        fruits.add("Apple");
        fruits.add("Orange");
        fruits.add("Banana");

        Collections.sort(fruits);

        int i=0;
        for(String temp: fruits){
                System.out.println("fruits " + ++i + " : " + temp);
        }
```

Output

```
fruits 1 : Apple
fruits 2 : Banana
fruits 3 : Orange
fruits 4 : Pineapple
```

3. Sort an Object with Comparable

How about a Java Object? Let create a Fruit class:

```
public class Fruit{

        private String fruitName;
        private String fruitDesc;
```

```
        private int quantity;

        public Fruit(String fruitName, String fruitDesc, int quantity) {
                super();
                this.fruitName = fruitName;
                this.fruitDesc = fruitDesc;
                this.quantity = quantity;
        }

        public String getFruitName() {
                return fruitName;
        }
        public void setFruitName(String fruitName) {
                this.fruitName = fruitName;
        }
        public String getFruitDesc() {
                return fruitDesc;
        }
        public void setFruitDesc(String fruitDesc) {
                this.fruitDesc = fruitDesc;
        }
        public int getQuantity() {
                return quantity;
        }
        public void setQuantity(int quantity) {
                this.quantity = quantity;
        }
}
```

To sort it, you may think of **Arrays.sort()** again, see below example :

```
package com.mkyong.common.action;

import java.util.Arrays;

public class SortFruitObject{

        public static void main(String args[]){

                Fruit[] fruits = new Fruit[4];

                Fruit pineappale = new Fruit("Pineapple", "Pineapple description",70);
                Fruit apple = new Fruit("Apple", "Apple description",100);
                Fruit orange = new Fruit("Orange", "Orange description",80);
                Fruit banana = new Fruit("Banana", "Banana description",90);

                fruits[0]=pineappale;
                fruits[1]=apple;
                fruits[2]=orange;
                fruits[3]=banana;

                Arrays.sort(fruits);

                int i=0;
                for(Fruit temp: fruits){
                   System.out.println("fruits " + ++i + " : " + temp.getFruitName() +
                        ", Quantity : " + temp.getQuantity());
                }

        }
}
```

Nice try, but, what you expect the **Arrays.sort()** will do? You didn't even mention what to sort in the Fruit class. So, it will hits the following error :

```
Exception in thread "main" java.lang.ClassCastException:
com.mkyong.common.Fruit cannot be cast to java.lang.Comparable
        at java.util.Arrays.mergeSort(Unknown Source)
        at java.util.Arrays.sort(Unknown Source)
```

To sort an Object by its property, you have to make the Object implement the **Comparable** interface and override the **compareTo()** method. Lets see the new Fruit class again.

```
public class Fruit implements Comparable<Fruit>{

	private String fruitName;
	private String fruitDesc;
	private int quantity;

	public Fruit(String fruitName, String fruitDesc, int quantity) {
		super();
		this.fruitName = fruitName;
		this.fruitDesc = fruitDesc;
		this.quantity = quantity;
	}

	public String getFruitName() {
		return fruitName;
	}
	public void setFruitName(String fruitName) {
		this.fruitName = fruitName;
	}
	public String getFruitDesc() {
		return fruitDesc;
	}
	public void setFruitDesc(String fruitDesc) {
		this.fruitDesc = fruitDesc;
	}
	public int getQuantity() {
		return quantity;
	}
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}

	public int compareTo(Fruit compareFruit) {

		int compareQuantity = ((Fruit) compareFruit).getQuantity();

		//ascending order
		return this.quantity - compareQuantity;

		//descending order
		//return compareQuantity - this.quantity;

	}
}
```

The new Fruit class implemented the **Comparable** interface, and overrided the **compareTo()** method to compare its quantity property in ascending order.

> The **compareTo()** method is hard to explain, in integer sorting, just remember
>
> 1. this.quantity – compareQuantity is ascending order.
> 2. compareQuantity – this.quantity is descending order.
>
> To understand more about compareTo() method, read this **Comparable documentation**.

Run it again, now the Fruits array is sort by its quantity in ascending order.

```
fruits 1 : Pineapple, Quantity : 70
fruits 2 : Orange, Quantity : 80
fruits 3 : Banana, Quantity : 90
fruits 4 : Apple, Quantity : 100
```

4. Sort an Object with Comparator

How about sorting with Fruit's "fruitName" or "Quantity"? The Comparable interface is only allow to sort a single property. To sort with multiple properties, you need **Comparator**. See the new updated Fruit class again :

```
import java.util.Comparator;

public class Fruit implements Comparable<Fruit>{

	private String fruitName;
	private String fruitDesc;
	private int quantity;

	public Fruit(String fruitName, String fruitDesc, int quantity) {
		super();
		this.fruitName = fruitName;
		this.fruitDesc = fruitDesc;
		this.quantity = quantity;
	}

	public String getFruitName() {
		return fruitName;
	}
	public void setFruitName(String fruitName) {
		this.fruitName = fruitName;
	}
	public String getFruitDesc() {
		return fruitDesc;
	}
	public void setFruitDesc(String fruitDesc) {
		this.fruitDesc = fruitDesc;
	}
	public int getQuantity() {
		return quantity;
	}
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}

	public int compareTo(Fruit compareFruit) {

		int compareQuantity = ((Fruit) compareFruit).getQuantity();

		//ascending order
		return this.quantity - compareQuantity;

		//descending order
		//return compareQuantity - this.quantity;

	}

	public static Comparator<Fruit> FruitNameComparator
			= new Comparator<Fruit>() {

	    public int compare(Fruit fruit1, Fruit fruit2) {

	      String fruitName1 = fruit1.getFruitName().toUpperCase();
	      String fruitName2 = fruit2.getFruitName().toUpperCase();

	      //ascending order
	      return fruitName1.compareTo(fruitName2);

	      //descending order
	      //return fruitName2.compareTo(fruitName1);
	    }

	};
}
```

The Fruit class contains a static **FruitNameComparator** method to compare the “fruitName”. Now the Fruit object is able to sort with either “quantity” or “fruitName” property. Run it again.

1. Sort Fruit array based on its “fruitName” property in ascending order.

```
Arrays.sort(fruits, Fruit.FruitNameComparator);
```

Output

```
fruits 1 : Apple, Quantity : 100
```

```
fruits 2 : Banana, Quantity : 90
fruits 3 : Orange, Quantity : 80
fruits 4 : Pineapple, Quantity : 70
```

2. Sort Fruit array based on its “quantity” property in ascending order.

```
Arrays.sort(fruits)
```

Output

```
fruits 1 : Pineapple, Quantity : 70
fruits 2 : Orange, Quantity : 80
fruits 3 : Banana, Quantity : 90
fruits 4 : Apple, Quantity : 100
```

The **java.lang.Comparable** and **java.util.Comparator** are powerful but take time to understand and make use of it, may be it’s due to the lacking of detail example.

My thoughts…

In future, Arrays class should provides more generic and handy method – **Arrays.sort(Object, String, flag)**.

To sort a object array by its “fruitName” in ascending order.

```
Arrays.sort(fruits, fruitName, Arrays.ASCENDING);
```

To sort a object array by its “quantity” in ascending order.

```
Arrays.sort(fruits, quantity, Arrays.DESCENDING);
```

Reference

1. **Comparable documentation**
2. **Comparator documentation**

Tags : java | sorting

About the Author

mkyong

Founder of **Mkyong.com** and **HostingCompass.com**, love Java and open source stuff. Follow him on **Twitter**, or befriend him on **Facebook** or **Google Plus**. If you like my tutorials, consider make a donation to **these charities**.

Comments

Martin · a year ago

I think it's not this

Arrays.sort(fruits, Fruit.FruitNameComparator);

but this

Collections.sort(fruits, Fruit.FruitNameComparator);

14 · Reply · Share ›

pink freud Martin · 5 days ago

fruits is an Array, not a List.

Reply · Share ›

lee Martin · 4 months ago

isn't collections used when using vector/arrayList?

Reply · Share ›

Balaram · 2 years ago

why super() is used in Fruit class(no.3) ?

9 · Reply · Share ›

Shishir Biyyala · a year ago

Thanks @mkyong.

One comment : The statement, "The Fruit class contains a static FruitNameComparator method to compare the “fruitName”" is wrong. FruitNameComparator is a static instance variable of type Comparator in Fruit class. Also, I think it's not mandatory for it to exist as a static member under Fruit, as this could be an equivalent:

```
class FruitComparator implements Comparator<fruit> {
public int compare(Fruit f1, Fruit f2) {
return (f1.getFruitName().toUpperCase()).compareTo((f2.getFruitName().toUpperCase()));
}
}
```

Arrays.sort(fruits, new FruitNameComparator()); would be the call in the main method.

5 · Reply · Share ›

divya Shishir Biyyala · 10 months ago

yes u r right.....

.....

Reply · Share ›

Ram divya · 7 months ago

thats why mk said ClassCastException occurence

people you look on clear way.....

Reply · Share ›

Victor Gil Shishir Biyyala · a year ago

You are right Shishir, I also noticed the wrong sentence, and I totally agree with your alternative implementation proposal.

Reply · Share ›

java code Victor Gil · 10 months ago

I idea of making comparator static inner class is design. if its outside the class then its out of sight.

Reply · Share ›

Mark · 2 years ago

your implementation of comparator is very advanced so I didn't understand it, however comparable was ok. could you please simplify it for me a total beginner to understand? If you don't mind send your simplified version of comparator to assan_shab@yahoo.com

Thanks for help.

2 · Reply · Share ›

Prem Kumar Mark · 2 years ago

Comparator is easy to understand but sometimes a bit confusing because of the difficult to grab documentation from oracle or earlier Sun. It is one of the most important things you need to know while working with java.

Try reading it here http://javahash.com/java-compa...

Reply · Share ›

cprasad Prem Kumar · 2 years ago

That link was excellent thanks Prem kumar.
1 · Reply · Share ›

Arul ➦ Prem Kumar · a year ago
thank u
Reply · Share ›

exex zian · a year ago
I seriously appreciate your thought on `Arrays.sort(Object, String, flag). .` it looks better this way
1 · Reply · Share ›

Mario Moreno · a year ago
Your articles are fantastic! Thanks
1 · Reply · Share ›

Constantin · a year ago
Here's an open source library to sort multiple columns in a delimited string: https://sourceforge.net/projec...
1 · Reply · Share ›

Ramesh · 2 years ago
what is the need of hascode&equals override..

the simple examle.
1 · Reply · Share ›

paul · 2 years ago
Very clear example. Thank you very much.
1 · Reply · Share ›

Sridhar.Goranti · 22 days ago
Thank you mkyong. It was very useful :)
Reply · Share ›

Mahadev · a month ago
would like to know...internally what sorting technique is used when we use comparable or comparator to compare objects in collection
Reply · Share ›

kuku · a month ago
thank you for sharing this tutorial , it is great :)
Reply · Share ›

Andrea · 2 months ago
Thank you very much, your guide was really helpful
Reply · Share ›

Vivek Pal · 2 months ago
To the point explanation
Thanks
Reply · Share ›

Chris · 2 months ago
Thank so very much! I used this to great value in a project where I needed to sort Edges of a graph by their numeric label, and later on I needed to topologically sort the whole graph, so I defined a comparator that sorted by postvisit time. Works like a charm.
Reply · Share ›

Ramtin · 4 months ago
Shouldn't it be: public int compareTo(Object fruit){...} in order it inherits the compareTo-method?
Reply · Share ›

JJ · 5 months ago
This class implements the Comparator interface. You should consider whether or not it should also implement the Serializable interface. If a comparator is used to construct an ordered collection such as a TreeMap, then the TreeMap will be serializable only if the comparator is also serializable. As most comparators have little or no state, making them serializable is generally easy and good defensive programming.
findbugs SE_COMPARATOR_SHOULD_BE_SERIALIZABLE
Reply · Share ›

nisha · 5 months ago

why super() is used in Fruit class(no.3) ?

Reply · Share ›

Rahul · 6 months ago

Where to write main method in this program and how to call

Arrays.sort(fruits, fruitName, Arrays.ASCENDING); I tried but it is giving compile time error

Reply · Share ›

Milind Durugkar · 6 months ago

CompareTo method not required in Fruit Class. As you are using String.CompareTo(String) not object.ComparTo(Object).

Reply · Share ›

Ram · 7 months ago

nice though ha...

Reply · Share ›

Yannyezixin · 7 months ago

Thanks! It help me!

Reply · Share ›

na · 8 months ago

made many stops here for reference. i love this write up. when i forget, i will just drop by here for reference. great writeup.

Reply · Share ›

Gautham · 8 months ago

To compare with Strings, we can use the compareToIgnoreCase() method which returns a positive, negative or a 0 depending upon whether the first string has a greater, lesser or the same character as that of the second string. It is done through lexicographical comparison which means that the compareToIgnoreCase() method compares character by character starting from the left and as soon as it reaches a character which is different, it returns the corresponding values as stated above.

Here's the code snippet of the overridden compareTo() method,
public int compareTo(Object obj)
{
Fruit fruit=(Fruit)obj;
return this.fruitName.compareToIgnoreCase(fruit.fruitName);
}

Reply · Share ›

feliciafay · 8 months ago

Thank you for clarifying the reason to use comparable and comparator. It is better to know why they are needed before the details of how to write the code. Thanks a lot!

Reply · Share ›

Gustavo Stork · 8 months ago

Thank you so much. I understand very quickly.

Reply · Share ›

Beckers Ben · 10 months ago

Very good read, thank you!

Reply · Share ›

LiTTle · a year ago

When I want to sort an array of objects the object should implement the Comparable! If I want to compare using the Comparator should my class implements Comparable again?

Reply · Share ›

Nam Nguyen · a year ago

if i don' t know how many objects, i think would use List<fruit> replacement for array[]

have the differences between use list<> and arrary[] ???

Reply · Share ›

anand · a year ago

Awesome..!!! a simple and clear Explanation...just want to add one doesn't need to implement Comparable Interface if they only want to use Comparator...

thanks Man..:))

Reply • Share ›

jizhask · 2 years ago

excellent, big thanks!

Reply • Share ›

Santiago · 2 years ago

Man you rock!!!...

Reply • Share ›

Rai · 2 years ago

big thanks!!

Reply • Share ›

Ahmed · 2 years ago

thank you very much ..

that was really helpful :)

Reply • Share ›

Andres Bernal · 2 years ago

very well explained, thanks

Reply • Share ›

Debayan Das · 2 years ago

Thanks, nicely explained!

Reply • Share ›

Mohamed Ennahdi El Idrissi · 2 years ago

I am just curious about the static fields used in the anonymous class instantiation: Should we put those static fields in a different class?
Or should they remain in the bean?

Thanks!

Reply • Share ›

Dilip Singh · 2 years ago

This is very useful for me thank u very much.....

Reply • Share ›

Simon · 2 years ago

your Arrays.sort(fruits, fruitName, Arrays.ASCENDING); doesn't make any sense, unless it is "fruitName" which is very unJava-like. Java 8 or 9 should provide lambdas, though to lessen the need for tons of ridiculously short classes such as Comparators.

Reply • Share ›

Parag · 2 years ago

I think your code (for comparator) is somewhat wrong. The class whose objects are to be sorted using compare() must implement Comparator, but u've only implemented Comparable.

Reply • Share ›

Alexander · 2 years ago

>In future, Arrays class should provides more generic and handy method ? Arrays.sort(Object, String, flag).
OMG, desc is a mirrored asc, sort it asc, and mirror.

Reply • Share ›

Developer Links

Android Getting Started
Google App Engine – Java
Spring 2.5.x Documentation
Spring 3.2.x Documentation
Spring 4.1.x Documentation
Java EE 5 Tutorial
Java EE 6 Tutorial
Java EE 7 Tutorial
Java 6 API
Java 7 API
Java 8 API
JSF Home Page
JSP Home Page
Maven Central Repository
Hibernate ORM
JAX-WS Home Page
JAX-RS Home Page (Jersey)

Friends & Partners

Java Code Geeks
TestNG Founder
DZone

Build Tools

Apache Ant
Apache Maven
Gradle

About Mkyong.com

Mkyong.com is for Java and J2EE developers, all examples are simple and easy to understand, of course it is well tested in my development environment.

Mkyong.com is created, written by, and maintained by Yong Mook Kim, aka Mkyong. It is built on **WordPress**, hosted by **Liquid Web**, and the caches are served by CloudFlare CDN.