

Industrial strength Java/JEE Career Companion to open more doors

[Home](#)
[Java FAQs](#)
[600+ Java Q&As](#)
[Career](#)
[Tutorials](#)
[Member](#)
[Why?](#)
[Can u Debug?](#)
[Java 8 ready?](#)
[Top X](#)
[Productivity Tools](#)
[Judging Experience?](#)

[Home](#) › [Interview](#) › [Core Java Interview Q&A](#) › [Collection and Data structures](#) ›

Java 8: Different ways to sort a collection of objects in pre and post Java 8

Java 8: Different ways to sort a collection of objects in pre and post Java 8

Posted on November 8, 2014 by Arulkumaran Kumaraswamipillai — No

[Comments](#) ↓

The object we are going to sort is a *Person*.

```

1 public class Person {
2
3     public enum Gender {FEMALE, MALE};
4
5     private String name;
6     private Integer age;
7     private Gender gender;
8
9     public Person(String name, Integer age, Gender
10         this.name = name;
11         this.age = age;
12         this.gender = gender;
13     }
14
15     //getter, setter, equals(...), and hashCode() m

```

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[Java Overview \(4\)](#)

[Data types \(6\)](#)

[constructors-methc](#)

[Reserved Key Wor](#)

[Classes \(3\)](#)

[Objects \(8\)](#)

[OOP \(10\)](#)

[GC \(2\)](#)

[Generics \(5\)](#)

[FP \(8\)](#)

[IO \(7\)](#)

[Multithreading \(12\)](#)

[Algorithms \(5\)](#)

[Annotations \(2\)](#)

[Collection and Data](#)

♦ Find the first n

♦ Java Collector

♥ Java Iterable \

♥♦ HashMap & H

```

16
17 @Override
18 public String toString() {
19     return "Person [name=" + name + ", age=" + age
20 }
21 }
22

```

Option 1: Writing your own **Comparator** implementation.

This can be done as an anonymous inner class instead of a separate class.

```

1 import java.util.Comparator;
2
3 public class PersonComparator implements Compara
4
5 @Override
6 public int compare(Person o1, Person o2) {
7
8     //by gender first
9     int i1 = o1.getGender().compareTo(o2.getGender
10     if (i1 != 0) return i1;
11
12     //by name next
13     int i2 = o1.getName().compareTo(o2.getName())
14     if (i2 != 0) return i2;
15
16     //by age
17     return o1.getAge().compareTo(o2.getAge());
18 }
19 }
20

```

The test class

```

1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class PersonTest {
5
6     public static void main(String[] args) {
7
8         List<Person> people = new ArrayList<>();
9         people.add(new Person("John", 35, Person.Gende
10         people.add(new Person("John", 32, Person.Gende
11         people.add(new Person("Simone", 30, Person.Gen
12         people.add(new Person("Shawn", 30, Person.Gend
13
14         System.out.println("before sorting = " + peop
15         people.sort(new PersonComparator());
16         System.out.println("after sorting = " + peopl
17
18     }
19 }
20

```

◆ Sorting objects

02: ◆ Java 8 Stre

04: Understandir

4 Java Collection

If Java did not ha

Java 8: Different

Part-3: Java Tre

Sorting a Map by

When to use whi

⊞ Differences Betwe

⊞ Event Driven Progr

⊞ Exceptions (2)

⊞ Java 7 (2)

⊞ Java 8 (24)

⊞ JVM (6)

⊞ Reactive Programn

⊞ Swing & AWT (2)

⊞ JEE Interview Q&A (3

⊞ Pressed for time? Jav

⊞ SQL, XML, UML, JSC

⊞ Hadoop & BigData Int

⊞ Java Architecture Inte

⊞ Scala Interview Q&As

⊞ Spring, Hibernate, & I

⊞ Testing & Profiling/Sa

⊞ Other Interview Q&A 1

⊞ 📺 Free Java Interview

16 Technical Key Areas

open all | close all

⊞ Best Practice (6)

⊞ Coding (26)

⊞ Concurrency (6)

⊞ Design Concepts (7)

⊞ Design Patterns (11)

⊞ Exception Handling (3

⊞ Java Debugging (21)

⊞ Judging Experience I

Option 2: The Option 1 is not bad, but the the moment you need to handle null element values, the *PersonComparator* will have more code. One of the best practices in Java is “Don’t reinvent the wheel”. So, let’s use the *BeanComparator*, *NullComparator*, and *ComparatorChain* from the **Apache commons library** commons-beanutils - > commons-beanutils-bean-collections that uses reflection. The example below also handles null values.

```

1 import java.util.ArrayList;
2 import java.util.List;
3
4 import org.apache.commons.beanutils.BeanComparat
5 import org.apache.commons.collections.comparator
6 import org.apache.commons.collections.comparator
7
8 public class PersonTest {
9
10  public static void main(String[] args) {
11
12      List<Person> people = new ArrayList<>();
13      people.add(new Person("John", 35, Person.Gende
14      people.add(new Person("John", 32, Person.Gende
15      people.add(new Person("Simone", 30, Person.Gen
16      people.add(new Person("Shawn", 30, Person.Gend
17      people.add(new Person("Shawn", 30, null));
18
19      System.out.println("before sorting = " + peop
20
21      //Apache commons-beanutils.commons-beanutils-b
22      ComparatorChain comparatorChain = new Comparat
23      //null is compared s lower
24      comparatorChain.addComparator(new BeanComparat
25      //null is compared as higher
26      comparatorChain.addComparator(new BeanComparat
27      comparatorChain.addComparator(new BeanComparat
28      people.sort(comparatorChain);
29
30      System.out.println("after sorting = " + peopl
31
32  }
33 }
34

```

Option 3: Using the **Google Gauva library** to sort the collection in functional programming style.

```

1 import java.util.ArrayList;

```

- ⊞ [Low Latency \(7\)](#)
- ⊞ [Memory Managemen](#)
- ⊞ [Performance \(13\)](#)
- ⊞ [QoS \(8\)](#)
- ⊞ [Scalability \(4\)](#)
- ⊞ [SDLC \(6\)](#)
- ⊞ [Security \(13\)](#)
- ⊞ [Transaction Managen](#)

80+ step by step Java Tutorials

[open all](#) | [close all](#)

- ⊞ [Setting up Tutorial \(6\)](#)
- ⊞ [Tutorial - Diagnosis \(2](#)
- ⊞ [Akka Tutorial \(9\)](#)
- ⊞ [Core Java Tutorials \(2](#)
- ⊞ [Hadoop & Spark Tuto](#)
- ⊞ [JEE Tutorials \(19\)](#)
- ⊞ [Scala Tutorials \(1\)](#)
- ⊞ [Spring & Hibernate Ti](#)
- ⊞ [Tools Tutorials \(19\)](#)
- ⊞ [Other Tutorials \(45\)](#)

100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- ⊞ [Can you write code? \(](#)
- ⊞ [♦ Complete the given](#)
- ⊞ [Converting from A to I](#)
- ⊞ [Designing your classe](#)
- ⊞ [Java Data Structures](#)
- ⊞ [Passing the unit tests](#)
- ⊞ [What is wrong with th](#)
- ⊞ [Writing Code Home A](#)
- ⊞ [Written Test Core Jav](#)

```

2 import java.util.Comparator;
3 import java.util.List;
4
5 import com.google.common.collect.ComparisonChain
6 import com.google.common.collect.Ordering;
7
8 public class PersonTest {
9
10 public static void main(String[] args) {
11
12     List<Person> people = new ArrayList<>();
13     people.add(new Person("John", 35, Person.Gender.Male));
14     people.add(new Person("John", 32, Person.Gender.Female));
15     people.add(new Person("Simone", 30, Person.Gender.Female));
16     people.add(new Person("Shawn", 30, Person.Gender.Male));
17     people.add(new Person("Shawn", 30, null));
18
19     System.out.println("before sorting = " + people);
20
21     //anonymous inner class using the Google Guava
22     people.sort(new Comparator<Person>() {
23
24         @Override
25         public int compare(Person o1, Person o2) {
26             return ComparisonChain.start()
27                 .compare(o1.getGender(), o2.getGender())
28                 .compare(o1.getName(), o2.getName(), Ordering.natural())
29                 .compare(o1.getAge(), o2.getAge(), Ordering.natural())
30                 .result();
31         }
32     });
33
34     System.out.println("after sorting = " + people);
35 }
36 }
37 }
38 }
39 }
40

```

[Written Test JEE \(1\)](#)

How good are your?

[open all](#) | [close all](#)
[Career Making Know-](#)
[Job Hunting & Resum](#)

Option 4: If you are using **Java 8**, using the **functional programming** approach.

```

1 import java.util.ArrayList;
2 import java.util.Comparator;
3 import java.util.List;
4
5 public class PersonTest {
6
7     public static void main(String[] args) {
8
9         List<Person> people = new ArrayList<>();
10        people.add(new Person("John", 35, Person.Gender.Male));
11        people.add(new Person("John", 32, Person.Gender.Female));
12        people.add(new Person("Simone", 30, Person.Gender.Female));
13        people.add(new Person("Shawn", 30, Person.Gender.Male));
14        people.add(new Person("Shawn", 30, null));
15
16        System.out.println("before sorting = " + people);
17
18        //java 8 approach fro multi-fields
19        Comparator<Person> multiFieldComparator =

```

```

20         Comparator.comparing(Person::getGender)
21                     .thenComparing(Person::getName)
22                     .thenComparing(Person::getAge)
23
24     people.sort(multiFieldComparator);
25     System.out.println("after sorting = " + people);
26
27 }
28 }
29

```

Option 5: If you are using **Java 8**, using **parallel processing**. Very similar to option 4, but processed in parallel with minor changes.

```

1  import java.util.ArrayList;
2  import java.util.Comparator;
3  import java.util.List;
4  import java.util.stream.Collectors;
5
6  public class PersonTest {
7
8      public static void main(String[] args) {
9
10         List<Person> people = new ArrayList<>();
11         people.add(new Person("John", 35, Person.Gender.MALE));
12         people.add(new Person("John", 32, Person.Gender.MALE));
13         people.add(new Person("Simone", 30, Person.Gender.FEMALE));
14         people.add(new Person("Shawn", 30, Person.Gender.FEMALE));
15         people.add(new Person("Shawn", 30, null));
16
17         System.out.println("before sorting = " + people);
18
19         Comparator<Person> multiFieldComparator =
20             Comparator.comparing(Person::getGender)
21                       .thenComparing(Person::getName)
22                       .thenComparing(Person::getAge)
23
24         //parallel() processing using Fork/Join
25         List<Object> sortedPeople = people.stream()
26                                         .parallel()
27                                         .sorted(multiFieldComparator)
28                                         .collect(Collectors.toList());
29
30         System.out.println("after sorting = " + sortedPeople);
31
32     }
33 }
34

```

Output:

before sorting = [Person [name=John, age=35, gender=MALE], Person [name=John, age=32, gender=MALE], Person [name=Simone, age=30, gender=FEMALE], Person [name=Shawn, age=30, gender=FEMALE], Person [name=Shawn, age=30, gender=null]]

```
gender=MALE], Person [name=Shawn, age=30,  
gender=null]]
```

```
after sorting = [Person [name=Shawn, age=30, gender=null],  
Person [name=Simone, age=30, gender=FEMALE], Person  
[name=John, age=32, gender=MALE], Person [name=John,  
age=35, gender=MALE], Person [name=Shawn, age=30,  
gender=MALE]]
```

Popular Posts

♦ 11 Spring boot interview questions & answers

823 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

765 views

18 Java scenarios based interview Questions and Answers

399 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views



Arulkumaran Kumaraswamipillai



Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ Top 6 Java 8 features you can start using now

Java 7: Top 8 new features with examples ▶

Posted in Collection and Data structures, Java 8, member-paid

Leave a Reply

Logged in as geethika. [Log out?](#)

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)
☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy

or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.