

Java Stream sorted()

📅 Last Updated: March 15, 2022 👤 By: Lokesh Gupta 📁 Java 8 💎 Java 8, Java Sorting, Java Stream Basics, Java Stream Methods

Learn to use **Stream sorted()** method to [sort](#) the elements in a Stream by their natural order. We can also apply custom orders on the elements using the provided [Comparator](#).

Table Of Contents

[1. Stream sort\(\) Method](#)

[1.1. Stream sorted\(\)](#)

[1.2. Stream sorted\(comparator\)](#)

[2. Stream sorted\(\) Examples](#)

[Example 1: Sorting in Natural Order](#)

[Example 2: Reverse Ordering](#)

[Example 3: Custom Ordering using Comparator](#)

[Example 4: Sorting using Lambda Expressions](#)

1. Stream sort() Method

The [Stream](#) interface provides two methods for sorting the Stream elements.

- **sorted()** – Provides the default sorting
- **sorted(Comparator)** – Sorting based on provided comparator.

1.1. Stream sorted()

Syntax

```
Stream<T> sorted()
```

- **sorted()** is a **stateful intermediate operation** which returns a new Stream.
- It returns a stream consisting of the elements of this stream, sorted according to **natural order**.
- If the elements of this stream are not **Comparable**, a **java.lang.ClassCastException** may be thrown when the terminal operation is executed.
- For ordered streams, the sort is stable.
- For unordered streams, no stability guarantees are made.

1.2. Stream sorted(comparator)

Syntax

```
Stream<T> sorted(Comparator<? super T> comparator)
```

- This is a **stateful intermediate operation** which returns a new stream.
- It returns a stream consisting of the elements of this stream, sorted according to the provided Comparator..
- For ordered streams, the sort is stable.
- For unordered streams, no stability guarantees are made.

2. Stream sorted() Examples

Example 1: Sorting in Natural Order

In the given Java example, we are sorting a stream of integers in the natural order and printing them into the standard output.

```
List<Integer> list = Arrays.asList(2, 4, 1, 3, 7, 5, 9, 6, 8);

List<Integer> sortedList = list.stream()
    .sorted()
    .collect(Collectors.toList());

System.out.println(sortedList);
```

Program output.

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Example 2: Reverse Ordering

In the given Java example, we are sorting a stream of integers in *reverse order* using a Comparator and printing them into the standard output.

```
List<Integer> list = Arrays.asList(2, 4, 1, 3, 7, 5, 9, 6, 8);

List<Integer> sortedList = list.stream()
    .sorted(Comparator.reverseOrder())
    .collect(Collectors.toList());

System.out.println(sortedList);
```

Program output.

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Example 3: Custom Ordering using Comparator

In the given Java example, we are sorting a stream of integers in *reverse order* using a custom *Comparator*.

```
List<Integer> list = Arrays.asList(2, 4, 1, 3, 7, 5, 9, 6, 8);

Comparator<Integer> reverseComparator = new Comparator<Integer>() {
    @Override
    public int compare(Integer i1, Integer i2) {
        return i2.compareTo(i1);
    }
};

List<Integer> sortedList = list.stream()
    .sorted(reverseComparator)
    .collect(Collectors.toList());

System.out.println(sortedList);
```

Program output.

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Example 4: Sorting using Lambda Expressions

Java example to sort a stream of integers in reverse order using [lambda expression](#) to specify the comparison logic.

We are rewriting the previous Comparator logic with an inline lambda expression.

```
List<Integer> list = Arrays.asList(2, 4, 1, 3, 7, 5, 9, 6, 8);

List<Integer> sortedList = list.stream()
    .sorted( (i1, i2) -> i2.compareTo(i1) )
    .collect(Collectors.toList());

System.out.println(sortedList);
```

Program output.

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Drop me your questions related to **Stream sorted() example** in Java Stream API.

Happy Learning !!

[Sourcecode on Github](#)

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. [Checking if an Array is Sorted in Java](#)
2. [Java Stream reuse – traverse stream multiple times?](#)
3. [Java Stream toArray\(\)](#)
4. [Java Stream findFirst\(\)](#)
5. [Java Stream findAny\(\)](#)
6. [Java Stream count\(\) Matches with filter\(\)](#)
7. [Java Stream max\(\)](#)
8. [Java Stream limit\(\)](#)
9. [Java Stream skip\(\)](#)
0. [Sorting a Stream by Multiple Fields in Java](#)

Join 7000+ Awesome Developers

Get the latest updates from industry, awesome resources, blog updates and much more.

Email Address

Subscribe

** We do not spam !!*

1 thought on “Java Stream sorted()”

prashanth

December 11, 2019 at 4:46 pm

Hi,i have a requirement of id,name and age ...i want to sort only age....kindly provide me the answer

[Reply](#)

Leave a Comment

Name *

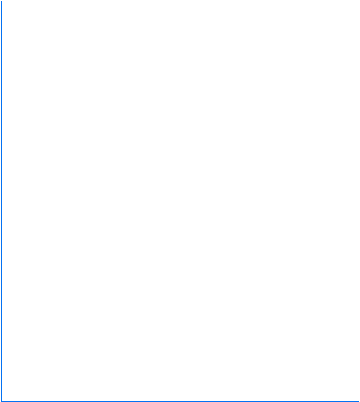
Email *

Website

☐ Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment





HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- [About Me](#)
- [Contact Us](#)
- [Privacy policy](#)
- [Advertise](#)
- [Guest Posts](#)

Blogs

[REST API Tutorial](#)



Copyright © 2022 · Hosted on [Cloudways](#) · [Sitemap](#)