HowToDoInJava

Java Stream skip()

Last Updated: March 30, 2022 | By: Lokesh Gupta | Java 8 | Java Stream Basics, Java Stream Methods

Stream *skip(n)* method is used to **skip the first `'n'` elements from the given Stream**.

The `skip()` method returns a new Stream consisting of the remaining elements of the original Stream, after the specified **n** elements have been discarded in the **encounter order**.

Table Of Contents

1. Stream skip() Method

1.1. Method Syntax

Syntax

```
Stream<T> skip(long n)
```

The **n** is the number of leading elements to be discarded. It returns a new *Stream* consisting of elements picked from the original stream.

The method may throw **IllegalArgumentException** if **n** is negative.

1.2. Description

- **Stream skip()** method is stateful **intermediate operation**. Stateful operations, such as distinct and sorted, may incorporate state from previously seen elements when processing new elements.
- Returns a stream consisting of the remaining elements of the stream after discarding the first n elements of the stream.
- If the stream contains fewer than n elements then an empty stream will be returned.
- Generally `skip()` is a cheap operation, it can be quite expensive on ordered parallel pipelines, especially for large values of n.
- Using an unordered stream source (such as `generate(Supplier)`) or removing the ordering constraint with `BaseStream.unordered()` may result in significant speedups of `skip()` in parallel pipelines.
- `skip()` skips the first n elements in the encounter order.

2. Stream skip() Example

In this Java program, we are using the `skip()` method to skip the first 5 even numbers from an infinite stream of even numbers and then collect the next 10 even numbers into a new Stream.

Skip 5 and then collect 10 numbers

```
Stream<Integer> evenNumInfiniteStream = Stream.iterate(0, n -> n + 2)

List<Integer> newList = evenNumInfiniteStream
        .skip(5)
```

```
        .limit(10)
        .collect(Collectors.toList());

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

Program output.

```
Console

[10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

3. Conclusion

The Stream *skip()* method can be **useful in certain cases where we need to get the elements from a Stream but first, we need to skip a few elements from the Stream**.

The fact, that `skip()` returns the elements in the encounter order, makes it very useful for normal business usecases as well.

Happy Learning !!

Sourcecode on Github

Recommended Reading:

1. Java Stream reuse – traverse stream multiple times?
2. Java Stream count() Matches with filter()
3. Java Stream forEach()
4. Java Stream sorted()
5. Java Stream max()
6. Java Stream min()
7. Java Stream peek()
8. Java Stream limit()
9. Java Stream findFirst()
0. Java Stream findAny()

Leave a Comment

Name *

Email *

Website

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

Post Comment

Search ...