HowToDoInJava

Java Stream limit()

📅 Last Updated: March 30, 2022   👤 By: Lokesh Gupta   📁 Java 8   🏷️ Java Stream Basics, Java Stream Methods

Stream limit(*n*) is used to retrieve a number of elements from the Stream while the count must not be greater than *n*. The `limit()` method **returns a new Stream** consisting of the elements of the given stream, truncated to be no longer than `maxSize` in length.

1. Stream limit() Method

1.1. Syntax

Syntax

```
Stream<T> limit(long maxSize)
```

Here `maxSize` the number of elements the stream should be limited to; and the method return value is a new `Stream` consisting of elements picked from the original stream.

1.2. Description

- **Stream.limit()** method is [short-circuiting](#) **intermediate operation**. An intermediate operation is short-circuiting if, when presented with infinite input, it may produce a finite stream as a result. Please note that a terminal operation is short-circuiting if, when presented with infinite input, it may terminate in finite time.

- It returns a stream consisting of the maximum elements, no longer than given size in length, of current stream.

- Generally, `limit()` is cheap operation but may sometimes be expensive if `maxSize` has a large value and stream is parallely processed.

- Using an unordered stream source (such as `generate(Supplier)`) or removing the ordering constraint with [`BaseStream.unordered()`](#) may result in significant speedups of `limit()` in parallel pipelines.

- `limit()` returns the first n elements in the encounter order.

2. Stream limit() Examples

Example 1: Getting first 10 even numbers from an infinite stream of even numbers

In the given below example, we are creating an infinite stream using `iterate()` method. Then we are taking the first 10 even numbers using the method `limit(10)`.
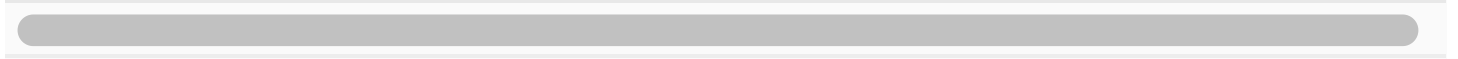
Finally, we are collecting the even numbers from the stream into a `List` using `collect(Collectors.toList())` method.

Collect first 10 even numbers

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

List<Integer> newList = evenNumInfiniteStream.limit(10)
                                        .collect(Collectors.toList());

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

Program output.

Output

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

3. Difference between *skip()* and *limit()*

- The `limit(N)` method returns first **N** elements in the **encounter order** of the Stream.

- The `skip(N)` discards the first N elements of the Stream.

```
List<Integer> list = Stream.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    .skip(6)
    .collect(Collectors.toList());

System.out.println(newList);    //[7, 8, 9, 10]
```

4. Conclusion

Java 8 Stream `limit()` method can be **useful in certain cases where we need to get the elements from a stream and the count of elements will be determined at runtime**.

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

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. [Java Regex to limit the number of words in input](#)

2. [Java Regex to limit the number of lines in text](#)

3. [How to Increase Console Output Limit in Eclipse](#)

4. [Java Stream reuse – traverse stream multiple times?](#)

5. [Java Stream forEach()](#)

6. [Java Stream max()](#)

7. [Java Stream min()](#)

8. [Java Stream skip()](#)

9. [Java Stream toArray()](#)

0. [Java Stream findAny()](#)

_____

Join 7000+ Awesome Developers

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

Email Address

Subscribe

* We do not spam !!

Leave a Comment

Name *

Email *

Website

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

Post Comment

Search … 🔍

A message from our sponsor

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

REST API Tutorial