Difference Between map() And flatMap() In Java Stream

Last Updated : 14 Sep, 2022

In Java, the Stream interface has a map() and flatmap() methods and both have intermediate stream operation and return another stream as method output. Both of the functions map() and flatMap are used for transformation and mapping operations. map() function produces one output for one input value, whereas flatMap() function produces an arbitrary no of values as output (ie zero or more than zero) for each input value.

The Syntax of the map() is represented as:

```
<R> Stream<R> map(Function<? super T, ? extends R> mapper)
```

The Syntax of the flatMap() is represented as:-

```
<R> Stream<R> flatMap(Function<? super T, ? extends Stream<? extends
R>> mapper)
```

Where R is the element type of the new stream. The stream is an interface and T is the type of stream elements and mapper is a stateless function that is applied to each element and the function returns the new stream.

map() can be used where we have to map the elements of a particular collection to a certain function, and then we need to return the stream which contains the updated results.

Example: Multiplying All the elements of the list by 3 and returning the updated list.

flatMap() can be used where we have to flatten or transform out the string, as we cannot flatten our string using map().

Example: Getting the 1st Character of all the String present in a List of Strings and returning the result in form of a stream.

Difference Between map() and flatmap()

map()	flatMap()
The function passed to map() operation returns a single value for a single input.	The function you pass to flatmap() operation returns an arbitrary number of values as the output.
One-to-one mapping occurs in map().	One-to-many mapping occurs in flatMap().
Only perform the mapping.	Perform mapping as well as flattening.
Produce a stream of value.	Produce a stream of stream value.
map() is used only for transformation.	flatMap() is used both for transformation and mapping.

Below are the Java Programs using map() function:

Java

```java
// Java program using map() function
import java.io.*;
import java.util.*;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
class GFG {

    public static void main(String[] args)
    {
        // making the array list object
        ArrayList<String> fruit = new ArrayList<>();
        fruit.add("Apple");
        fruit.add("mango");
        fruit.add("pineapple");
        fruit.add("kiwi");
        System.out.println("List of fruit-" + fruit);

        // lets use map() to convert list of fruit
        List list = fruit.stream()
                        .map(s -> s.length())
                        .collect(Collectors.toList());
        System.out.println("List generated by map-" + list);
    }
}
```

Output:

```
List of fruit-[Apple, mango, pineapple, kiwi]
List generated by map-[5, 5, 9, 4]
```

Below is the Java Program using flatMap():

Java

```
// Java program using flatMap() function
import java.io.*;
import java.util.*;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
class GFG {
    public static void main(String[] args)
    {
        // making the arraylist object of List of Integer
        List<List<Integer> > number = new ArrayList<>();

        // adding the elements to number arraylist
        number.add(Arrays.asList(1, 2));
        number.add(Arrays.asList(3, 4));
        number.add(Arrays.asList(5, 6));
        number.add(Arrays.asList(7, 8));

        System.out.println("List of list-" + number);

        // using flatmap() to flatten this list
        List<Integer> flatList
            = number.stream()
                  .flatMap(list -> list.stream())
                  .collect(Collectors.toList());

        // printing the list
        System.out.println("List generate by flatMap-"
                           + flatList);
    }
}
```

Output

```
List of list-[[1, 2], [3, 4], [5, 6], [7, 8]]
List generate by flatMap-[1, 2, 3, 4, 5, 6, 7, 8]
```

T tejsw... + Follow 20

Previous Article

Intermediate Methods of Stream in Java

Next Article

Array to Stream in Java

Similar Reads

Stream flatMap() in Java with examples

Stream flatMap(Function mapper) returns a stream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying...

4 min read

Difference between Stream.of() and Arrays.stream() method in Java

Arrays.stream() The stream(T[] array) method of Arrays class in Java, is used to get a Sequential Stream from the array passed as the parameter with its elements. It returns a...

5 min read

IntStream flatMap(IntFunction mapper) in Java

IntStream flatMap(IntFunction mapper) returns a stream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced b...

2 min read

Character Stream Vs Byte Stream in Java

A stream is a sequence of data. I/O Stream refers to a stream that is unlikely a method to sequentially access a file. I/O Stream means an input source or output destination...

4 min read

Java Program to Convert String to Char Stream Without Using Stream

Char stream defines the array of characters. In this article, we will learn the different types of methods for converting a String into a char stream in Java without using Stream. Let u...

4 min read

View More Articles

Article Tags : java-stream Technical Scripter 2020 Difference Between Java +2 More

Practice Tags : Java

Company

About Us
Legal
Careers
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning Tutorial
ML Maths
Data Visualisation Tutorial
Pandas Tutorial
NumPy Tutorial
NLP Tutorial
Deep Learning Tutorial

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Git
AWS
Docker
Kubernetes
Azure

Explore

Hack-A-Thons
GfG Weekly Contest
DSA in JAVA/C++
Master System Design
Master CP
GeeksforGeeks Videos
Geeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

HTML & CSS

HTML
CSS
Web Templates
CSS Frameworks
Bootstrap
Tailwind CSS
SASS
LESS
Web Design

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

Competitive Programming

Top DS or Algo for CP
Top 50 Tree
Top 50 Graph
Top 50 Array
Top 50 String

GCP

DevOps Roadmap

System Design

High Level Design

Low Level Design

UML Diagrams

Interview Guide

Design Patterns

OOAD

System Design Bootcamp

Interview Questions

Preparation Corner

Company-Wise Recruitment Process

Resume Templates

Aptitude Preparation

Puzzles

Company-Wise Preparation

Management & Finance

Management

HR Management

Finance

Organisational Behaviour

Marketing

More Tutorials

Software Development

Software Testing

Product Management

SEO - Search Engine Optimization

Linux

Excel

All Cheatsheets

Top 50 DP

Top 15 Websites for CP

JavaScript

JavaScript Examples

TypeScript

ReactJS

NextJS

AngularJS

NodeJS

Lodash

Web Browser

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

World GK

Free Online Tools

Typing Test

Image Editor

Code Formatters

Code Converters

Currency Converter

Random Number Generator

Random Password Generator

GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects