

[JAVA Offline Classes](#)[Java Arrays](#)[Java Strings](#)[Java OOPs](#)[Java Collection](#)[Java 8 Tutorial](#)[Java Multithreading](#)[Java Exception Handling](#)[Java Programs](#) [Explore Our Geeks  
Community](#)[Write an Interview Experience](#)[Share Your Campus  
Experience](#)[Java Tutorial](#)[Java | How to start learning  
Java](#)[Java Programming Basics](#)[How JVM Works - JVM  
Architecture?](#)[Object Oriented  
Programming \(OOPs\) Concept  
in Java](#)[Classes and Objects in Java](#)[Java Constructors](#)

# Stream In Java

[Read](#)[Discuss](#)[Practice](#)

Introduced in Java 8, Stream API is used to process collections of objects. A stream in Java is a sequence of objects that supports various methods which can be pipelined to produce the desired result.

## Use of Stream in Java

*There uses of Stream in Java are mentioned below:*

- 1. Stream API is a way to express and process collections of objects.*
- 2. Enable us to perform operations like filtering, mapping, reducing and sorting.*

## How to Create Java Stream?

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our

[Cookie Policy](#) & [Privacy Policy](#)

**Got It !**

syntax given on how to declare Java Stream.

## Syntax

```
Stream<T> stream;
```

Here T is either a class, object, or data type depending upon the declaration.

### MERN Full Stack Web Development

As you're reading this, two more seats got filled in our MERN Classroom Program. Don't wait! Secure your spot now and dominate the MERN development world. [Reserve Seat](#)

## Java Stream Features

The features of Java stream are mentioned below:

- A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.

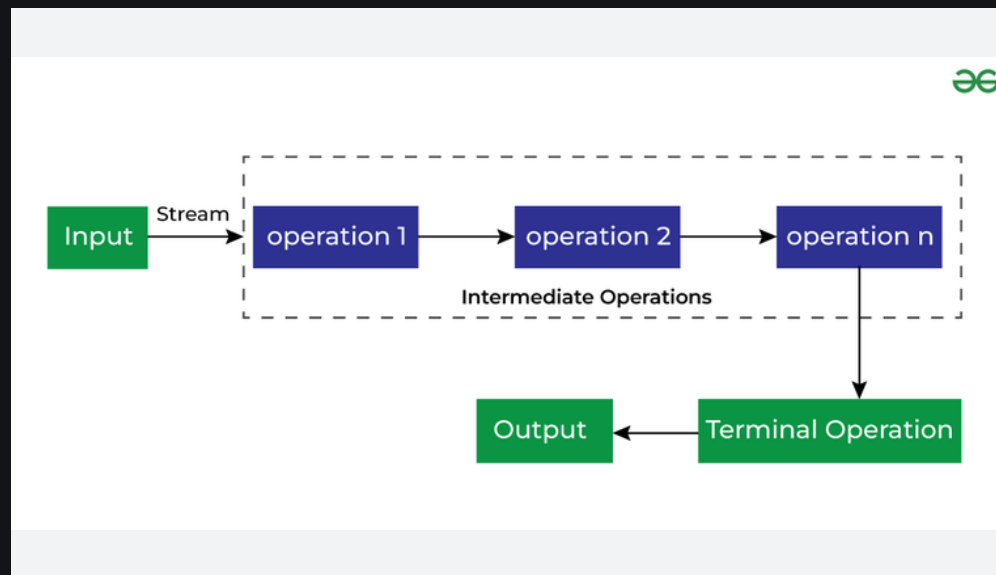
- Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

## Different Operations On Streams

There are two types of Operations in Streams:

1. Intermediate Operations
2. Terminate Operations

### Intermediate Operations



Intermediate Operations are the types of operations in which multiple methods are chained in a row.

1. Methods are chained together.
2. Intermediate operations transform a stream into another stream.
3. It enables the concept of filtering where one method filters data and passes it to another method after processing.

## Important Intermediate Operations

There are a few Intermediate Operations mentioned below:

### 1. map()

The map method is used to return a stream consisting of the results of applying the given function to the elements of this stream.

```
List number = Arrays.asList(2,3,4,5);  
List square = number.stream().map(x-  
>x*x).collect(Collectors.toList());
```

### 2. filter()

The filter method is used to select elements as per the Predicate passed as an argument.

```
List names =  
Arrays.asList("Reflection","Collection","Stream");  
List result = names.stream().filter(s-  
>s.startsWith("S")).collect(Collectors.toList());
```

The sorted method is used to sort the stream.

```
List names =  
Arrays.asList("Reflection","Collection","Stream");  
List result =  
names.stream().sorted().collect(Collectors.toList());
```

## Terminal Operations

Terminal Operations are the type of Operations that return the result. These Operations are not processed further just return a final result value.



## Important Terminal Operations

There are a few Terminal Operations mentioned below:

The collect method is used to return the result of the intermediate operations performed on the stream.

```
List number = Arrays.asList(2,3,4,5,3);  
Set square = number.stream().map(x-  
>x*x).collect(Collectors.toSet());
```

## 2. forEach()

The forEach method is used to iterate through every element of the stream.

```
List number = Arrays.asList(2,3,4,5);  
number.stream().map(x->x*x).forEach(y-  
>System.out.println(y));
```

## 3. reduce()

The reduce method is used to reduce the elements of a stream to a single value. The reduce method takes a BinaryOperator as a parameter.





```
List number = Arrays.asList(2,3,4,5);  
int even = number.stream().filter(x->x%2==0).reduce(0,  
(ans,i)-> ans+i);
```

Here ans variable is assigned 0 as the initial value and i is added to

***Note:** Intermediate Operations are running based on the concept of Lazy Evaluation, which ensures that every method returns a fixed value(Terminal operation) before moving to the next method.*

## Example of Java Stream

### Java



```
// Java program to demonstrate
// the use of stream in java
import java.util.*;
import java.util.stream.*;

class Demo {
    public static void main(String args[])
    {
        // create a list of integers
        List<Integer> number = Arrays.asList(2, 3, 4, 5);

        // demonstration of map method
        List<Integer> square
            = number.stream()
                .map(x -> x * x)
                .collect(Collectors.toList());

        // create a list of String
        List<String> names = Arrays.asList(
            "Reflection", "Collection", "Stream");
```

```

List<String> result
    = names.stream()
        .filter(s -> s.startsWith("S"))
        .collect(Collectors.toList());

System.out.println(result);

// demonstration of sorted method
List<String> show
    = names.stream()
        .sorted()
        .collect(Collectors.toList());

System.out.println(show);

// create a list of integers
List<Integer> numbers
    = Arrays.asList(2, 3, 4, 5, 2);

// collect method returns a set
Set<Integer> squareSet
    = numbers.stream()
        .map(x -> x * x)
        .collect(Collectors.toSet());

System.out.println(squareSet);

// demonstration of forEach method
number.stream()
    .map(x -> x * x)
    .forEach(y -> System.out.println(y));

// demonstration of reduce method
int even
    = number.stream()

```



```
        System.out.println(even);  
    }  
}
```

## Output

```
[4, 9, 16, 25]  
[Stream]  
[Collection, Reflection, Stream]  
[16, 4, 9, 25]  
4  
9  
16  
25  
6
```

## Important Points/Observations of Java Stream



1. A stream consists of a source followed by zero or more intermediate methods combined together (pipelined) and a terminal method to process the objects obtained from the source as per the methods described.
2. Stream is used to compute elements as per the pipelined methods without altering the original value of the object.

Feeling lost in the vast world of Backend Development? It's time for a change! Join our [Java Backend Development - Live Course](#) and embark on an exciting journey to master backend development efficiently and on schedule.

### What We Offer:

- Comprehensive Course
- Expert Guidance for Efficient Learning
- Hands-on Experience with Real-world Projects
- Proven Track Record with 100,000+ Successful Geeks

Last Updated : 08 Sep, 2023

 316 

[◀ Previous](#)

[Next ▶](#)

[Collections in Java](#)

[Exceptions in Java](#)

### Similar Reads

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

Character Stream Vs Byte Stream in Java

foreach() loop vs Stream  
foreach() vs Parallel Stream  
foreach()

Java Stream | Collectors  
toCollection() in Java

[Stream.reduce\(\) in Java with examples](#)

[Stream.concat\(\) in Java](#)

[Stream.distinct\(\) in Java](#)

[Stream sorted\(\) in Java](#)

## Complete Tutorials

[Java AWT Tutorial](#)

[Spring MVC Tutorial](#)

[Spring Tutorial](#)

[Spring Boot Tutorial](#)

[Java 8 Features - Complete Tutorial](#)



GeeksforGeeks

**Article Tags :** [Java - util package](#) , [java-stream](#) , [Java](#)

**Practice Tags :** [Java](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our

[Cookie Policy](#) & [Privacy Policy](#).

## Complete Data Analytics Program

Warning! Clicking here means you will not be able to stop yourself from mastering Data Analytics, unlocking unparalleled insights, and becoming #1 in the field. **Transform Now**



A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305



### Company

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

### Explore

Job-A-Thon  
Hiring Challenge  
Hack-A-Thon  
GfG Weekly Contest  
Offline Classes (Delhi/NCR)  
DSA in JAVA/C++  
Master System Design  
Master CP  
GeeksforGeeks

### Languages

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

### DSA

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

### Data Science & ML

Data Science With Python  
Data Science For Beginner  
Machine Learning Tutorial  
ML Maths  
Data Visualisation Tutorial

### HTML & CSS

HTML  
CSS  
Bootstrap  
Tailwind CSS  
SASS  
LESS  
Web Design

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our

[Cookie Policy](#) & [Privacy Policy](#)

# Deep Learning Tutorial

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

GATE CS Notes

Operating  
Systems

Computer  
Network

Database  
Management  
System

Software  
Engineering

Digital Logic  
Design

Engineering  
Maths

- Git
- AWS
- Docker
- Kubernetes
- Azure
- GCP
- DevOps
- Roadmap

- Top DS or Algo for CP
- Top 50 Tree
- Top 50 Graph
- Top 50 Array
- Top 50 String
- Top 50 DP
- Top 15 Websites for CP

What is System Design

Monolithic and Distributed SD

High Level Design or HLD

Low Level Design or LLD

Crack System Design Round

System Design Interview Questions

Grokking Modern System Design

- TypeScript
- ReactJS
- NextJS
- AngularJS
- NodeJS
- Express.js
- Lodash
- Web Browser

SSC/  
BANKING

Class 10	Chemistry	Indian	Income Tax	Geography	SBI PO Syllabus
Class 9	Biology	Economics	Finance	Notes	SBI Clerk
Class 8	Social Science	Macroeconomics	Economics	History Notes	Syllabus
Complete Study	English	Microeconomics		Science and	IBPS PO
Material	Grammar	Statistics for		Technology	Syllabus
		Economics		Notes	IBPS Clerk
				Economy Notes	Syllabus
				Ethics Notes	SSC CGL
				Previous Year	Practice Papers
				Papers	

Colleges	Companies	Preparation Corner	Exams	More Tutorials	Write & Earn
Indian Colleges Admission & Campus Experiences	IT Companies Software Development Companies	Company Wise Preparation Preparation for SDE Experienced Interviews Internship Interviews Competitive Programming Aptitude Preparation Puzzles	JEE Mains JEE Advanced GATE CS NEET UGC NET	Software Development Software Testing Product Management SAP SEO Linux Excel	Write an Article Improve an Article Pick Topics to Write Share your Experiences Internships
Top Engineering Colleges	Artificial Intelligence(AI) Companies				
Top BCA Colleges	CyberSecurity Companies				
Top MBA Colleges	Service Based Companies				
Top Architecture College	Product Based Companies				
Choose College For Graduation	PSUs for CS Engineers				

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved