

Industrial strength Java/JEE Career Companion to open more doors

search here ...

Go

Home

Java FAQs

600+ Java Q&amp;As

Career

Tutorials

Member

Why?

Can u Debug?

Java 8 ready?

Top X

Productivity Tools

Judging Experience?

[Home](#) › [Interview](#) › [Core Java Interview Q&A](#) › [Java 8](#) › Understanding Java 8

Streams and working with collections using Lambda expressions

# Understanding Java 8 Streams and working with collections using Lambda expressions

Posted on November 25, 2014 by Arulkumaran Kumaraswamipillai — No

[Comments](#) ↓

A stream is an infinite sequence of consumable elements (i.e a data structure) for the consumption of an operation or iteration. Any Collection can be exposed as a stream. The operations you perform on a stream can either be

- **intermediate** (map, filter, sorted, limit, skip, concat, substream, distinct, etc) producing another stream or
- **terminal** (forEach, reduce, collect, sum, max, count, matchAny, findFirst, findAny, etc) producing an object that is not a stream.

## 600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

- ✚ [Ice Breaker Interview](#)
- ✚ [Core Java Interview C](#)
- ✚ [Java Overview \(4\)](#)
- ✚ [Data types \(6\)](#)
- ✚ [constructors-methc](#)
- ✚ [Reserved Key Wor](#)
- ✚ [Classes \(3\)](#)
- ✚ [Objects \(8\)](#)
- ✚ [OOP \(10\)](#)
- ✚ [GC \(2\)](#)
- ✚ [Generics \(5\)](#)
- ✚ [FP \(8\)](#)
- ✚ [IO \(7\)](#)
- ✚ [Multithreading \(12\)](#)
- ✚ [Algorithms \(5\)](#)
- ✚ [Annotations \(2\)](#)
- ✚ [Collection and Dat](#)
- ✚ [Differences Betwee](#)
- ✚ [Event Driven Progr](#)
- ✚ [Exceptions \(2\)](#)
- ✚ [Java 7 \(2\)](#)

Basically, you are building a pipeline as in Unix

```
ls -l | grep "Dec" | Sort +4n | more
```

***stream()*** is a default method added to the *Collection* interface in Java 8. The *stream()* returns a *java.util.Stream* interface with multiple abstract methods like filter, map, sorted, collect, etc. *DelegatingStream* implements these abstract methods.

### Java 8 Example 1:

```

1 package com.java8.examples;
2 import java.math.BigDecimal;
3 import java.util.Arrays;
4 import java.util.List;
5 import java.util.stream.Collectors;
6
7
8 public class EmployeeTest {
9
10 private static List<Employee> employees = Arrays
11     .asList(
12         new Employee("Steve", BigDecimal.valueOf(35000)),
13         new Employee("Peter", BigDecimal.valueOf(65000)),
14         new Employee("Sam", BigDecimal.valueOf(75000)),
15         new Employee("John", BigDecimal.valueOf(25000))
16     );
17
18     public static void main(String[] args) {
19         //e is the parameter for Employee
20         List<Employee> fullTimeEmployees = employees
21             .stream()
22             .filter(e -> e.getWorkType() == Employee.WORK_TYPE_FULLTIME)
23             .collect(Collectors.toList()); // returns
24         fullTimeEmployees.forEach(e -> System.out.println(e));
25     }
26 }
27

```

The **output** is:

```
Employee [name=Peter, salary=65000,
workType=FULLTIME]
```

```
Employee [name=Sam, salary=75000, workType=FULLTIME]
```

The above example creates a new list of full time employees. The operations *.stream()*, *.filter()* create the **intermediate streams**, hence they are chained, and the *.collect* is the

### Java 8 (24)

- 01: ♦ 19 Java 8 I
- 02: ♦ Java 8 Stre
- 03: ♦ Functional
- 04: ♥♦ Top 6 tips
- 04: Convert Lists
- 04: Understandir
- 05: ♥ 7 Java FP
- 05: ♦ Finding the
- 06: ♥ Java 8 way
- 07: ♦ Java 8 API
- 08: ♦ Write code
- 10: ♦ Executors
- Fibonacci numb
- Java 8 String str
- Java 8 using the
- Java 8: 7 useful
- Java 8: Different
- Java 8: Does "O
- Java 8: What is c
- Learning to write
- Non-trivial Java 8
- Top 6 Java 8 fea
- Top 8 Java 8 fea
- Understanding J

### JVM (6)

### Reactive Programn

### Swing & AWT (2)

### JEE Interview Q&A (3

### Pressed for time? Jav

### SQL, XML, UML, JSC

### Hadoop & BigData Int

### Java Architecture Inte

### Scala Interview Q&As

### Spring, Hibernate, & I

### Testing & Profiling/Sa

### Other Interview Q&A 1

### Free Java Interview

**terminal operation** that returns the final List of full-time employees.

This is enabled in the *Iterable* interface, which is a functional interface with the *forEach* default method. A List and other collections implements the *Iterable* functional interface to allow lambda expressions.

The **Employee** class will look like

```

1 package com.java8.examples;
2 import java.math.BigDecimal;
3
4 public class Employee {
5
6     public enum WorkType {FULLTIME, PARTTIME, CASUAL}
7
8     private String name;
9     private BigDecimal salary;
10    private WorkType workType;
11
12    public Employee(String name, BigDecimal salary,
13        super();
14        this.name = name;
15        this.salary = salary;
16        this.workType = workType;
17    }
18
19    public String getName() {
20        return name;
21    }
22    public void setName(String name) {
23        this.name = name;
24    }
25    public BigDecimal getSalary() {
26        return salary;
27    }
28    public void setSalary(BigDecimal salary) {
29        this.salary = salary;
30    }
31    public WorkType getWorkType() {
32        return workType;
33    }
34    public void setWorkType(WorkType workType) {
35        this.workType = workType;
36    }
37
38    @Override
39    public String toString() {
40        return "Employee [name=" + name + ", salary="
41    }
42 }
43

```

## Java 8 Example 2:

## 16 Technical Key Areas

[open all](#) | [close all](#)

- [Best Practice \(6\)](#)
- [Coding \(26\)](#)
- [Concurrency \(6\)](#)
- [Design Concepts \(7\)](#)
- [Design Patterns \(11\)](#)
- [Exception Handling \(3\)](#)
- [Java Debugging \(21\)](#)
- [Judging Experience \(1\)](#)
- [Low Latency \(7\)](#)
- [Memory Management \(1\)](#)
- [Performance \(13\)](#)
- [QoS \(8\)](#)
- [Scalability \(4\)](#)
- [SDLC \(6\)](#)
- [Security \(13\)](#)
- [Transaction Management \(1\)](#)

## 80+ step by step Java Tutorials

[open all](#) | [close all](#)

- [Setting up Tutorial \(6\)](#)
- [Tutorial - Diagnosis \(2\)](#)
- [Akka Tutorial \(9\)](#)
- [Core Java Tutorials \(2\)](#)
- [Debugging Tutorial \(1\)](#)
- [Java 8 Tutorial \(4\)](#)
- [02: ♦ Java 8 Streams \(1\)](#)
- [Learning to write \(1\)](#)
- [Top 6 Java 8 features \(1\)](#)
- [Understanding J \(1\)](#)
- [JSON and Java Tutorial \(1\)](#)
- [XML and Java Tutorial \(1\)](#)
- [CSV and Java Tutorial \(1\)](#)
- [Hadoop & Spark Tutorial \(1\)](#)

Now, in the same example, if you want to print all the employees who earn more than 25,000, sorted by *WorkType*, in the format " earns on a basis.

```

1 //e is the parameter for Employee
2 employees.stream() //returns a stream (intermedi
3 .filter(e -> e.getSalary().doubleValue() > 25000
4 .sorted((e1,e2) -> e1.getWorkType().compareTo(e2
5 .map(e -> e.getName() + " earns " + e.getSalary(
6 .forEach(System.out::println); //terminal
7
8

```

**Isn't this cool?** This is like writing the **SQL where clause**.

**Output:**

Peter earns 65000 on a FULLTIME basis

Sam earns 75000 on a FULLTIME basis

Steve earns 35000 on a PARTTIME basis

## Java 8 Example 3:

Get a comma separated string of all employees earning more than 25,000.00 as salary.

```

1 //e is the parameter for Employee
2 String str = employees.stream() //returns a str
3 .filter(e -> e.getSalary().doubleValue() > 2500
4 .map(e -> e.getName())// returns a stream (inte
5 .distinct() //returns a stream (intermediate)
6 //terminal returning a string
7 .reduce("Distinct first names earning > 25000:")
8
9 System.out.println("CSV: " + str);
10

```

**Output:**

CSV: Distinct first names earning > 25000:;Steve,Peter,Sam

- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate T](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

## 100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(](#)
- [♦ Complete the given](#)
- [Converting from A to I](#)
- [Designing your classe](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with th](#)
- [Writing Code Home A](#)
- [Written Test Core Jav](#)
- [Written Test JEE \(1\)](#)

## How good are your .....?

[open all](#) | [close all](#)

- [Career Making Know-](#)
- [Job Hunting & Resum](#)

## Java 8 Example 4:

Aggregate the salary by *WorkType*.

```
1 //e is the parameter for Employee
2 employees.stream() //returns a stream (intermedi
3 .collect(Collectors.groupingBy(Employee::getWork
4 .forEach((gr,le) -> {System.out.println(" Aggreg
5 + le.stream().mapToDouble(e -> e.getSalary()).d
6 });
7
```

### Output:

Aggregated Salary for CASUAL is 25000.0  
Aggregated Salary for FULLTIME is 140000.0  
Aggregated Salary for PARTTIME is 35000.0

## Java 8 Example 5:

Get the employee who earns the highest salary.

```
1 //e is the parameter for Employee
2 Employee maxSalaryEmployee = employees.stream()
3 .max((e1, e2) -> (e1.getSalary().compareTo(e2.ge
4 .get());
5
6 System.out.println(maxSalaryEmployee);
7
```

### Output:

Employee [name=Sam, salary=75000, workType=FULLTIME]

## Popular Posts

♦ [11 Spring boot interview questions & answers](#)

825 views

♦ [Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers](#)

766 views

## 18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio

Latest Posts



### Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



**About** [Arulkumaran Kumaraswamipillai](#)

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](#) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ Learning to write functional programming with Java 8 with examples  
Java 8 using the Predicate functional interface ▶

**Posted in** Java 8, Java 8 Tutorial, member-paid

## Leave a Reply

[Logged in as geethika.](#) [Log out?](#)

### Comment

**Empowers you to open more doors, and fast-track**

### Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)  
☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

### Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

## Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

## © Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.