

Solving Real Time Queries Using Java 8 Features - Employee Management System

pramodbablad June 23, 2019 23

Last Updated : November 22nd, 2021

Let's try to solve some of the real time queries faced in the Employee Management System using Java 8 features.

We will be using following *Employee* class and *employeeList* as example while solving the queries.

1) *Employee* Class :

```
1  class Employee
2  {
3      int id;
4
5      String name;
6
7      int age;
8
9      String gender;
10
11     String department;
12
```

```
13     int yearOfJoining;  
14  
15     double salary;  
16  
17     public Employee(int  
18     {  
19         this.id = id;  
20         this.name = name;  
21         this.age = age;  
22         this.gender = gender;  
23         this.department = department;  
24         this.yearOfJoining = yearOfJoining;  
25         this.salary = salary;  
26     }  
27  
28     public int getId()  
29     {  
30         return id;  
31     }  
32  
33     public String getName()  
34     {  
35         return name;  
36     }  
37  
38     public int getAge()  
39     {  
40         return age;  
41     }  
42  
43     public String getGender()  
44     {  
45         return gender;  
46     }  
47  
48     public String getDepartment()  
49     {  
50         return department;  
51     }  
52  
53     public int getYearOfJoining()  
54     {  
55         return yearOfJoining;  
56     }  
57  
58     public double getSalary()  
59     {  
60         return salary;  
61     }  
62  
63     @Override  
64     public String toString()  
65     {  
66         return "Id : " + id + ", Name : " + name + ", Age : " + age + ", Gender : " + gender + ", Department : " + department + ", YearOfJoining : " + yearOfJoining + ", Salary : " + salary;  
67     }
```

```
68         +", age
69         +", Ger
70         +", Dep
71         +", Yea
72         +", Sal
73     }
74 }
```

2) List Of Employees : *employeeList*

```
1 List<Employee> employee
2
3 employeeList.add(new Er
4 employeeList.add(new Er
5 employeeList.add(new Er
6 employeeList.add(new Er
7 employeeList.add(new Er
8 employeeList.add(new Er
9 employeeList.add(new Er
10 employeeList.add(new Er
11 employeeList.add(new Er
12 employeeList.add(new Er
13 employeeList.add(new Er
14 employeeList.add(new Er
15 employeeList.add(new Er
16 employeeList.add(new Er
17 employeeList.add(new Er
18 employeeList.add(new Er
19 employeeList.add(new Er
```

Also Read : Java 8 Lambda Expressions

3) Real Time Queries On *employeeList*

- How many male and female employees are there in the organization?
- Print the name of all departments in the organization?
- What is the average age of male and female employees?
- Get the details of highest paid employee in the organization?
- Get the names of all employees who have joined after 2015?
- Count the number of employees in each department?
- What is the average salary of each department?
- Get the details of youngest male employee in the product development department?
- Who has the most working experience in the organization?
- How many male and female employees are there in the sales and marketing team?
- What is the average salary of male and female employees?
- List down the names of all employees in each department?
- What is the average salary and total salary of the whole organization?
- Separate the employees who are younger or equal to 25 years from those employees who are older than 25 years?
- Who is the oldest employee in the organization? What is his age and which department he belongs to?

Query 3.1 : How many male and female employees are there in the organization?

For queries such as above where you need to group the input elements, use the *Collectors.groupingBy()* method. In this query, we use *Collectors.groupingBy()* method which takes two arguments. We pass *Employee::getGender* as first argument which groups the input elements based on *gender* and *Collectors.counting()* as second argument which counts the number of entries in each group.

```
1 Map<String, Long> noOfMaleAndFemaleEmployeesInEachGender =
2 employeeList.stream().collect(Collectors.groupingBy(Employee::getGender,
3 Collectors.counting()));
4 System.out.println(noOfMaleAndFemaleEmployeesInEachGender);
```

Output :

```
{Male=11, Female=6}
```

Query 3.2 : Print the name of all departments in the organization?

Use *distinct()* method after calling *map(Employee::getDepartment)* on the stream. It will return unique departments.

```
1 employeeList.stream()  
2     .map(Employee::getDepartment)  
3     .distinct()  
4     .forEach(System.out::println);
```

Output :

HR
Sales And Marketing
Infrastructure
Product Development
Security And Transport
Account And Finance

Query 3.3 : What is the average age of male and female employees?

Use same method as query 3.1 but pass *Collectors.averagingInt(Employee::getAge)* as the second argument to *Collectors.groupingBy()*.

```
1 Map<String, Double> avgAge  
2 employeeList.stream().collect(Collectors.groupingBy(  
3     Employee::getGender, Collectors.averagingInt(Employee::getAge)  
4     System.out.println(avgAge);
```

Output :

{Male=30.181818181818183, Female=27.166666666666668}

Also Read : Java 8 Collectors

Query 3.4 : Get the details of highest paid employee in the organization?

Use *Collectors.maxBy()* method which returns maximum element wrapped in an *Optional* object based on supplied *Comparator*.

```
1 Optional<Employee> high
2 employeeList.stream().c
3
4 Employee highestPaidEmp
5
6 System.out.println("Det
7
8 System.out.println("===
9
10 System.out.println("ID
11
12 System.out.println("Nam
13
14 System.out.println("Age
15
16 System.out.println("Gen
17
18 System.out.println("Dep
19
20 System.out.println("Ye
21
22 System.out.println("Sal
```

Output :

Details Of Highest Paid Employee :

```
=====
ID : 277
Name : Anuj Chettiar
Age : 31
Gender : Male
Department : Product Development
Year Of Joining : 2012
Salary : 35700.0
```

Query 3.5 : Get the names of all employees who have joined after 2015?

For such queries which require filtering of input elements, use *Stream.filter()* method which filters input elements according to supplied *Predicate*.

```
1 employeeList.stream()  
2     .filter(e ->  
3     .map(Employee  
4     .forEach(Sys
```

Output :

Iqbal Hussain
Amelia Zoe
Nitin Joshi
Nicolus Den
Ali Baig

Query 3.6 : Count the number of employees in each department?

This query is same as query 3.1 but here we are grouping the elements by *department*.

```
1 Map<String, Long> employ  
2 employeeList.stream().cc  
3  
4 Set<Entry<String, Long>  
5  
6 for (Entry<String, Long>  
7 {  
8     System.out.println(  
9 }
```

Output :

Product Development : 5
Security And Transport : 2
Sales And Marketing : 3
Infrastructure : 3
HR : 2
Account And Finance : 2

Also Read : Java 8 Streams

Query 3.7 : What is the average salary of each department?

Use the same method as in the above query 3.6, but here pass *Collectors.averagingDouble(Employee::getSalary)* as second argument to *Collectors.groupingBy()* method.

```
1 Map<String, Double> avgSalary
2 employeeList.stream().collect(Collectors.groupingBy(
3     Employee::getDepartment,
4     Set<Entry<String, Double>>.toSet(),
5     Collectors.averagingDouble(Employee::getSalary)
6 for (Entry<String, Double> entry : avgSalary.entrySet())
7 {
8     System.out.println(entry.getKey() + " : " + entry.getValue());
9 }
```

Output :

```
Product Development : 31960.0
Security And Transport : 10750.25
Sales And Marketing : 11900.166666666666
Infrastructure : 15466.666666666666
HR : 23850.0
Account And Finance : 24150.0
```

Query 3.8 : Get the details of youngest male employee in the product development department?

For this query, use *Stream.filter()* method to filter male employees in product development department and to find youngest among them, use *Stream.min()* method.

```

1 Optional<Employee> your
2 employeeList.stream()
3     .filter(e -
4     .min(Compar
5
6 Employee youngestMaleEn
7
8 System.out.println("Det
9
10 System.out.println("---
11
12 System.out.println("ID
13
14 System.out.println("Nar
15
16 System.out.println("Age
17
18 System.out.println("Ye
19
20 System.out.println("Sa

```

Output :

Details Of Youngest Male Employee In Product Development :

ID : 222
 Name : Nitin Joshi
 Age : 25
 Year Of Joinging : 2016
 Salary : 28200.0

Query 3.9 : Who has the most working experience in the organization?

For this query, sort *employeeList* by *yearOfJoining* in natural order and first employee will have most working experience in the organization. To solve this query, we will be using *sorted()* and *findFirst()* methods of *Stream*.

```

1 Optional<Employee> seni
2 employeeList.stream().s
3
4 Employee seniorMostEmp
5

```

```
6 System.out.println("Ser
7
8 System.out.println("---
9
10 System.out.println("ID
11
12 System.out.println("Nar
13
14 System.out.println("Age
15
16 System.out.println("Ger
17
18 System.out.println("Age
19
20 System.out.println("Yea
21
22 System.out.println("Sal
```

Output :

Senior Most Employee Details :

ID : 177
Name : Manu Sharma
Age : 35
Gender : Male
Age : Account And Finance
Year Of Joinging : 2010
Salary : 27000.0

Also Read : Java 8 Optional Class

Query 3.10 : How many male and female employees are there in the sales and marketing team?

This query is same as query 3.1, but here use *filter()* method to filter sales and marketing employees.

```
1 Map<String, Long> countM
2 employeeList.stream()
3     .filter(e ->
4     .collect(Con
5
6 System.out.println(count
```

Output :

{Female=1, Male=2}

Query 3.11 : What is the average salary of male and female employees?

This query is same as query 3.3 where you have found average age of male and female employees. Here, we will be finding average salary of male and female employees.

```
1 Map<String, Double> avgSalary =
2   employeeList.stream().collect(Collectors.groupingBy(
3     Employee::getGender, Collectors.averagingDouble(Employee::getSalary)));
4 System.out.println(avgSalary);
```

Output :

{Male=21300.090909090908, Female=20850.0}

Query 3.12 : List down the names of all employees in each department?

For this query, we will be using *Collectors.groupingBy()* method by passing *Employee::getDepartment* as an argument.

```
1 Map<String, List<Employee>> employeesByDepartment =
2   employeeList.stream().collect(Collectors.groupingBy(
3     Employee::getDepartment, Collectors.toList()));
4 Set<Entry<String, List<Employee>>> entries = new HashSet<>();
5
6 for (Entry<String, List<Employee>> entry : employeesByDepartment.entrySet())
7 {
8     System.out.println("Department: " + entry.getKey());
9     System.out.println("Employees: ");
10    System.out.println("-----");
11    System.out.println(entry.getValue());
12    System.out.println("-----");
13    List<Employee> list = entry.getValue();
14
15    for (Employee e : list)
16        System.out.println(e.getName());
```

```
17 {  
18     System.out.pri  
19 }  
20 }
```

Output :

Employees In Product Development :

Murali Gowda
Wang Liu
Nitin Joshi
Sanvi Pandey
Anuj Chettiar

Employees In Security And Transport :

Iqbal Hussain
Jaden Dough

Employees In Sales And Marketing :

Paul Niksui
Amelia Zoe
Nicolus Den

Employees In Infrastructure :

Martin Theron
Jasna Kaur
Ali Baig

Employees In HR :

Jiya Brein
Nima Roy

Employees In Account And Finance :

Manu Sharma
Jyothi Reddy

Query 3.13 : What is the average salary and total salary of the whole organization?

For this query, we use *Collectors.summarizingDouble()* on *Employee::getSalary* which will return statistics of the employee salary like max, min, average and total.

```

1 DoubleSummaryStatistics
2 employeeList.stream().co
3
4 System.out.println("Aver
5
6 System.out.println("Tota

```

Output :

Average Salary = 21141.235294117647
Total Salary = 359401.0

Query 3.14 : Separate the employees who are younger or equal to 25 years from those employees who are older than 25 years.

For this query, we will be using *Collectors.partitioningBy()* method which separates input elements based on supplied *Predicate*.

```

1 Map<Boolean, List<Emple
2 employeeList.stream().c
3
4 Set<Entry<Boolean, List
5
6 for (Entry<Boolean, Lis
7 {
8     System.out.println(
9
10     if (entry.getKey())
11     {
12         System.out.pri
13     }
14     else
15     {
16         System.out.pri
17     }
18
19     System.out.println(
20
21     List<Employee> list
22
23     for (Employee e :
24     {
25         System.out.pri

```

```
26     }  
27 }
```

Output :

Employees younger than or equal to 25 years :

Paul Niksui
Amelia Zoe
Nitin Joshi
Nicolus Den
Ali Baig

Employees older than 25 years :

Jiya Brein
Martin Theron
Murali Gowda
Nima Roy
Iqbal Hussain
Manu Sharma
Wang Liu
Jaden Dough
Jasna Kaur
Jyothi Reddy
Sanvi Pandey
Anuj Chettiar

Query 3.15 : Who is the oldest employee in the organization? What is his age and which department he belongs to?

```
1 Optional<Employee> oldestEmployee  
2  
3 Employee oldestEmployee  
4  
5 System.out.println("Name: " + oldestEmployee.getName())  
6  
7 System.out.println("Age: " + oldestEmployee.getAge())  
8  
9 System.out.println("Department: " + oldestEmployee.getDepartment())
```

Output :

Name : Iqbal Hussain
Age : 43
Department : Security And Transport

References :

- Java 8 Splitter
 - Java 8 map() Vs flatMap()
 - Java 8 Stream Intermediate & Terminal Operation
 - Java 8 Collections Vs Streams
 - Java 8 Functional Interfaces
 - Java 8 merge two maps
 - Java 8 Sort HashMap By Keys
 - Java 8 Sort HashMap By Values
 - Java 8 StringJoiner
 - Java 8 Method References
 - Java 8 Interface changes
 - Java 8 Oracle Docs
-

Categories Java 8

Java 8
Collectors
Tutorial

23 Comments

Java
Singleton
Design
Pattern

Implementation

With
Examples**Pankaj**

September 6, 2019 (4:35 pm) #

Please give same king of real time example with different use cases

Reply

Abhishek Kumar

June 14, 2020 (2:00 am) #

Query 3.8 line no 3 , &&
please change && with &&

Reply

Karthik

August 29, 2020 (6:43 pm) #

if i want to find second largest salary in the list what i have to do?

Reply

Pankaj Lilhore

September 18, 2020 (7:59 pm) #

```
Optional maxSal=  
employeeList.stream().max(Comparator.comparingDouble(Employee::getSalary));
```

```
Optional sec=employeeList.stream().filter( s -> s.getSalary() !=  
maxSal.get().getSalary()).max(Comparator.comparing(Employee::getSalary));  
System.out.println("sec sal:"+sec.get());
```

Reply

Arvind

October 26, 2021 (10:37 pm) #

```
Optionalemp=emptiest.stream().sorted(comparingDouble(Employee::  
getSalary).reverse()).skip(1).findFirst();
```

Reply

harish

January 9, 2021 (6:51 pm) #

i need to find 1st three employee details based on highest salary?

[Reply](#)**ravi**

May 26, 2021 (3:13 pm) #

```
employeeList.stream().sorted(comparingDouble(Employee::getSalary).reversed()).limit(3).forEach(System.out::println);
```

[Reply](#)**harish**

January 10, 2021 (10:35 am) #

if i have an employee details is in text file how can we apply these examples to text file, please give me these examples by using text files.

[Reply](#)**Satish Kumar**

February 11, 2021 (8:42 pm) #

Very useful , I was looking for this for a long time.

[Reply](#)**Shivraj Singh**

May 24, 2021 (11:29 am) #

Thanks Sir Very help full.

[Reply](#)**Pooja Almiya**

June 2, 2021 (12:52 am) #

Very useful

[Reply](#)

neha

June 24, 2021 (1:41 pm) #

Create simple and small Employee class and write codes to filter all Female Employees working in CSE department from list of Employees by using stream.

Reply

Sameer

August 12, 2021 (7:18 pm) #

Very Useful. Thanks Alot

Reply

Rupendra Raghu

August 29, 2021 (12:39 pm) #

As per my understanding for string comparison we should use .equals method not ==.

Reply

AJAY LAMKHADE

September 1, 2021 (10:37 pm) #

No better example can be imagined than this one for such a topic, very well constructed. thanks a lot for making it simpler.

Reply

Amit Kumar

October 2, 2021 (9:26 pm) #

Great Explain No Second Thought

Reply

Gajendra Singh

October 11, 2021 (11:45 am) #

Thank you so much very helpful.

Reply

KONDA RAGHU

October 12, 2021 (12:18 am) #

How to find highest salary in each department

Reply

Soumya Mukherjee

October 27, 2021 (8:10 am) #

```
Map<Integer, Optional> empWithMaxSalaryDeptWise =  
employeeList.stream().collect(Collectors.groupingBy(Employee::getDeptId,  
Collectors.reducing(BinaryOperator.maxBy(Comparator.comparing(Employee::getSalary)))));  
  
empWithMaxSalaryDeptWise.entrySet().forEach(entry->  
System.out.println(entry.getKey()+"-----"+entry.getValue().get().getSalary()));
```

Reply

Vivek Manhar

November 4, 2021 (3:41 pm) #

How to write below code

1. List of Student has id, subject and marks need to fetch highest marks in each subject;
2. Map of College Name and StudentName List
filter out all college names starting with Letter "S" and output list of students who belong to that college.
also

Reply

Avinash Jayakar

November 17, 2021 (11:27 am) #

Exceptional Content .. Thanks a lot !!

Reply

shankar s.

January 27, 2022 (2:27 pm) #

Thanks a lot. very very useful.

Reply

Victor

April 5, 2022 (1:48 am) #

Hello,

What if you want to return your key to be name if the department and your value to be the salary for the department ?

I can't get the key like this but I don't know how to continue to get the salary per position

```
Map getDeoNameAndSalary(List employeeList){  
    return employeeList.stream().map(Employee  
        ::getDepartment).collect(Collectors.groupingby(Function.identity()),// I know here I should take  
        the value but I cannot figure it out :((();
```