Java 8 Collectors Tutorial

pramodbablad June 7, 2019 2

Last Updated : June 27th, 2019

Java 8 Collectors tutorial mainly consist of three things – *Stream.collect()* method, *Collector* interface and *Collectors* class. *collect()* method is a terminal operation in *Stream* interface. *Collector* is an interface in *java.util.stream* package. *Collectors* class, also a member of *java.util.stream* package, is an utility class containing many static methods which perform some common reduction operations. Let's discuss them one by one.

1) *Stream.collect()* Method

collect() method is a terminal operation in *Stream* interface. It is a special case of reduction operation called mutable reduction operation because it returns mutable result container such as *List*, *Set* or *Map* according to supplied *Collector*.

```
import java.util.Arrays
import java.util.List;
import java.util.strear

public class Collectors
{
    public static void
    {
        List<Integer> r

        //collect() me

        List<Integer> c

        System.out.prir

        //OUTPUT : [5,
    }
}
```

2) *java.util.stream.Collector* Interface

java.util.stream.Collector interface contains four functions that work together to accumulate input elements into a mutable result container and optionally performs a final transformation on the result. Those four functions are,

a) *Supplier()* :

A function that creates and returns a new mutable result container.

b) *accumulator()* :

A function that accumulates a value into a mutable result container.

c) *combiner()* :

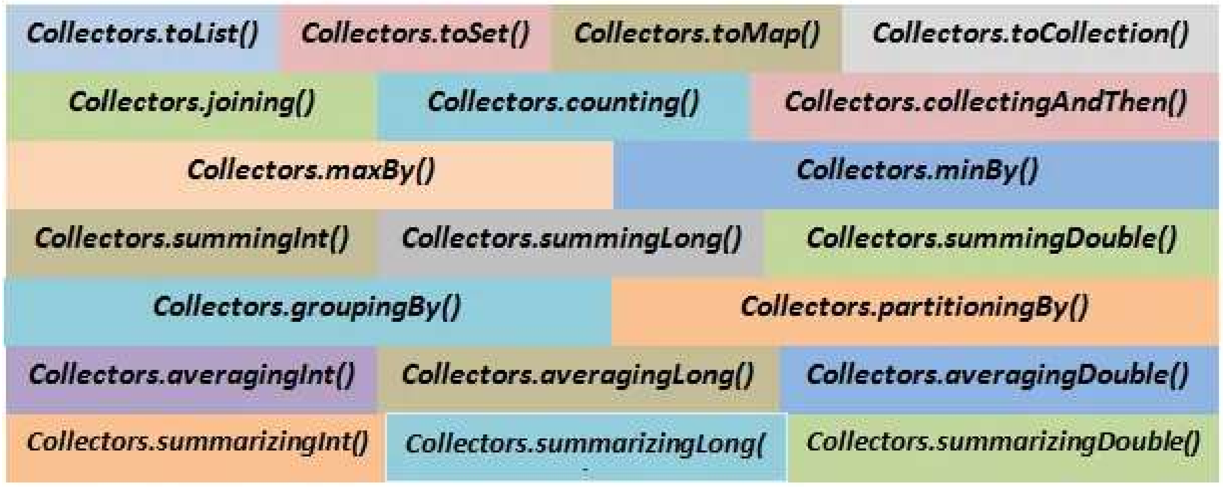
A function that accepts two partial results and merges them.

d) *finisher()* :

A function that performs final transformation from the intermediate accumulation type to the final result type.

3) *java.util.stream.Collectors* Class

java.util.stream.Collectors class contains static factory methods which perform some common reduction operations such as accumulating elements into Collection, finding min, max, average, sum of elements etc. All the methods of *Collectors* class return *Collector* type which will be supplied to *collect()* method as an argument.



Collectors.toList()	Collectors.toSet()	Collectors.toMap()	Collectors.toCollection()
Collectors.joining()	Collectors.counting()	Collectors.collectingAndThen()	
Collectors.maxBy()	Collectors.minBy()		
Collectors.summingInt()	Collectors.summingLong()	Collectors.summingDouble()	
Collectors.groupingBy()	Collectors.partitioningBy()		
Collectors.averagingInt()	Collectors.averagingLong()	Collectors.averagingDouble()	
Collectors.summarizingInt()	Collectors.summarizingLong(	Collectors.summarizingDouble()	

Let's see *Collectors* class methods one by one.

In the below coding examples, we will be using following *Student* class and *studentList*.

***Student* Class :**

```
class Student
{
    String name;

    int id;

    String subject;

    double percentage;

    public Student(Stri
    {
        this.name = nan
        this.id = id;
        this.subject =
        this.percentage
    }

    public String getNa
    {
        return name;
    }

    public int getId()
    {
        return id;
    }

    public String getSu
    {
        return subject;
    }

    public double getPe
    {
        return percent
    }

    @Override
    public String toStr
    {
        return name+"-'
    }
}
```

studentList **:**

```
List<Student> studentLi

studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
studentList.add(new Stu
```

3.1) *Collectors.toList()* :

It returns a *Collector* which collects all input elements into a new *List*.

Example : Collecting top 3 performing students into *List*

```
List<Student> top3Studer

System.out.println(top3S

//Output :

//[Vijay-19-Mathematics-
```

3.2) *Collectors.toSet()* :

It returns a *Collector* which collects all input elements into a new *Set*.

Example : Collecting subjects offered into *Set*.

```
Set<String> subjects = 

System.out.println(subje

//Output :

//[Economics, Literature
```

3.3) *Collectors.toMap()* :

This method returns a *Collector* which collects input elements into a *Map* whose keys and values are the result of applying mapping functions to input elements.

Example : Collecting name and percentage of each student into a *Map*

```
Map<String, Double> name

System.out.println(nameP

//Output :

//{Asif=89.4, Vijay=92.
```

3.4) *Collectors.toCollection()* :

This method returns a *Collector* which collects all input elements into a new *Collection*.

Example : Collecting first 3 students into *LinkedList*

```
LinkedList<Student> stu
System.out.println(stude
//Output :
//[Paul-11-Economics-78.
```

3.5) *Collectors.joining()* :

This method returns a *Collector* which concatenates input elements separated by the specified delimiter.

Example : Collecting the names of all students joined as a string

```
String namesJoined = stu
System.out.println(names
//Output :
//Paul, Zevin, Harish, 
```

3.6) *Collectors.counting()* :

It returns a *Collector* that counts number of input elements.

Example : Counting number of students.

```
Long studentCount = stu
System.out.println(stude
//Output : 10
```

3.7) ***Collectors.maxBy()*** :

This method returns a *Collector* that collects largest element in a stream according to supplied *Comparator*.

Example : Collecting highest percentage.

```
Optional<Double> highPer

System.out.println(highP

//Output : Optional[92.8
```

3.8) ***Collectors.minBy()*** :

This method returns a *Collector* which collects smallest element in a stream according to supplied *Comparator*.

Example : Collecting lowest percentage.

```
Optional<Double> lowPerc

System.out.println(lowPe

//Output : Optional[71.5
```

3.9) ***summingInt(), summingLong(), summingDouble()***

These methods returns a *Collector* which collects sum of all input elements.

Example : Collecting sum of percentages

```
Double sumOfPercentages

System.out.println(sumOf

//Output : 815.0
```

3.10) ***averagingInt(), averagingLong(), averagingDouble()***

These methods return a *Collector* which collects average of input elements.

Example : Collecting average percentage

```
Double averagePercentage

System.out.println(avera

//Output : 81.5
```

3.11) *summarizingInt(), summarizingLong(), summarizingDouble()*

These methods return a special class called *Int/Long/ DoubleSummaryStatistics* which contain statistical information like sum, max, min, average etc of input elements.

Example : Extracting highest, lowest and average of percentage of students

```
DoubleSummaryStatistics

System.out.println("Hig

System.out.println("Low

System.out.println("Ave

//Output :

//Highest Percentage :
//Lowest Percentage : 7
//Average Percentage :
```

3.12) *Collectors.groupingBy()* :

This method groups the input elements according supplied classifier and returns the results in a *Map*.

Example : Grouping the students by subject

```
Map<String, List<Studer

System.out.println(stuc

//Output :

```

```
//{Economics=[Paul-11-E
// Literature=[Xiano-14
// Computer Science=[Ze
// Mathematics=[Asif-16
// History=[Harish-13-H
```

3.13) *Collectors.partitioningBy()* :

This method partitions the input elements according to supplied *Predicate* and returns a *Map<Boolean, List<T>>*. Under the *true* key, you will find elements which match given *Predicate* and under the *false* key, you will find the elements which doesn't match given *Predicate*.

Example : Partitioning the students who got above 80.0% from who don't.

```
Map<Boolean, List<Studer

System.out.println(stude

//Output :

// {false=[Paul-11-Econo
//  true=[Zevin-12-Compu
```

3.14) *Collectors.collectingAndThen()* :

This is a special method which lets you to perform one more action on the result after collecting the result.

Example : Collecting first three students into *List* and making it unmodifiable

```
List<Student> first3Stud

System.out.println(first

//Output :

//[Paul-11-Economics-78.
```

Related Java 8 Tutorials :