HowToDoInJava

Using 'if-else' Conditions with Java Streams

Last Updated: March 3, 2022 By: Lokesh Gupta Java 8 if else, Java 8, Java Stream Basics

Learn to use the **if-else conditions logic** using Java Stream API to filter the items from a collection based on certain conditions.

Table Of Contents

1. The '`if-else`' Condition as `Consumer` Implementation

The `'if-else'` condition can be applied as a lambda expression in **forEach()** function in form of a Consumer action.

Consumer is a functional interface whose functional method is '`void accept(Object)`'. It represents an operation that accepts a single input argument and returns no result.

In the given example, we are checking ***if*** *a number is even then print a message,* ***else*** *print another message* for an odd number.

```
ArrayList<Integer> numberList
    = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5, 6));

Consumer<Integer> action = i -> {
    if (i % 2 == 0) {
```

```
            System.out.println("Even number :: " + i); //Or any other use
        } else {
            System.out.println("Odd  number :: " + i);  //Or any other us
        }
    };

    numberList.stream()
        .forEach(action);
```

- We can perform any kind of operation on the stream items instead of just printing the items to the console, e.g. storing the items to two separate lists or passing the items to other method calls.
- We can write as many *if-else* statements as required.
- We can also write the pass the Consumer implementation as an inline lambda expression to the `forEach()` function.

```
Arrays.asList(-1, 1, -2, 3, 4, -5, 6, 0).stream()
    .forEach(
        i -> {
          if (i == 0) {
            System.out.println("Number is 0");
          } else if (i > 0) {
            System.out.println("Positive Number");
          } else {
            System.out.println("Negative Number");
          }
        }
    );
```

2. The 'if' Condition with Predicates

If we intend to apply **only 'if' logic** then we can pass the condition directly do the `filter()` function as a Predicate.

In the given example, we are checking *if* a number is an even number then printing a message.

```
ArrayList<Integer> numberList = new ArrayList<>(Arrays.asList(1,2,3,4


Predicate<Integer> isEven = i -> i % 2 == 0;

numberList.stream()
    .filter(isEven)
    .forEach(System.out::println);
```

Using one of the above given two methods, we can apply any combination of **if-else conditions in Java 8 stream** elements.

Happy Learning !!

Sourcecode on Github

Recommended Reading:

1. Applying Multiple Conditions on Java Streams
2. Python if…else

3. Java if-else Statement
4. Boxed Streams in Java
5. Creating Infinite Streams in Java
6. Sorting Streams in Java
7. Finding Max and Min from List using Streams
8. Java Streams API
9. Creating Streams in Java
0. Primitive Type Streams in Java

2 thoughts on “Using ‘if-else’ Conditions with Java Streams”

a=i

September 25, 2019 at 5:06 pm

```
int a;
list.forEach(i-> {
if(i%2==0)
a=i;
})
```

how to handle this ? variable in lambda should be final or effectively final

Reply

Akash Gaba

March 15, 2020 at 2:15 pm

Multiple ways to this:

1. use AtomicInteger instead of int.

```
AtomicInteger a = new AtomicInteger(0);
list.forEach(i-> {
if(i%2==0)
a.set(i);
});
int answer = a.get();
```

2. Refactor: you are looking for last element divisible by 2.

```
int a = list.stream().filter(i -> i%2 == 0).reduce((first,
second) -> second).orElse(null);
```

Reply

Leave a Comment

Name *

Email *

Website

Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search ...

HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial