HowToDoInJava

Java Stream allMatch()

Last Updated: March 30, 2022 By: Lokesh Gupta Java 8 Java Stream Basics, Java Stream Methods

Java **Stream *allMatch()*** is a short-circuiting terminal operation that is used **to check if all the elements in the stream satisfy the provided predicate**.

Table Of Contents

1. Stream *allMatch()* Method

1.1. Syntax

Syntax

```
boolean allMatch(Predicate<? super T> predicate)
```

Here `predicate` a non-interfering, stateless predicate to apply to all the elements of the stream.

The `allMatch()` method returns always a `true` or `false`, based on the result of the evaluation.

1.2. Description

- It is a short-circuiting **terminal operation**.
- It returns whether all elements of this stream match the provided `predicate`.
- May not evaluate the `predicate` on all elements if not necessary for determining the result. Method returns `true` if all stream elements match the given predicate, else method returns `false`.
- If the stream is empty then `true` is returned and the predicate is not evaluated.
- The **difference between allMatch() and anyMatch()** is that `anyMatch()` returns `true` if any of the elements in a stream matches the given predicate. When using `allMatch()`, all the elements must match the given predicate.

2. Stream *allMatch()* Examples

Let us look at a few examples of `allMatch()` menthod to understand its usage.

Example 1: Checking if Any Element Contains Numeric Characters

In the given example, none of the strings in the Stream contain any numeric character. The `allMatch()` checks this condition in all the strings and finally returns `true`.

Checking all elements in the stream

```
Stream<String> stream = Stream.of("one", "two", "three", "four");

Predicate<String> containsDigit = s -> s.contains("\\d+") == false;
```

```
boolean match = stream.allMatch(containsDigit);

System.out.println(match);
```

Program output.

Output

```
true
```

Example 2: `Stream.allMatch()` with Multiple Conditions

To satisfy multiple conditions, create a composed predicate with two or more simple predicates.

In the given example, we have a list of `Employee`. We want to check if all the employees who are above the age of 50 – are earning above 40,000.

In the list, the employee `"B"` is earning below 40k and his age is above 50, so the result is `false`.

allMatch() with composed predicate

```
import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;
import java.util.stream.Stream;
import lombok.AllArgsConstructor;
import lombok.Data;

public class Main
{
  public static void main(String[] args)
  {
    Predicate<Employee> olderThan50 = e -> e.getAge() > 50;
    Predicate<Employee> earningMoreThan40K = e -> e.getSalary() > 40_
    Predicate<Employee> combinedCondition = olderThan50.and(earningMo

    boolean result = getEmployeeStream().allMatch(combinedCondition);
    System.out.println(result);
  }

  private static Stream<Employee> getEmployeeStream()
  {
    List<Employee> empList = new ArrayList<>();
    empList.add(new Employee(1, "A", 46, 30000));
    empList.add(new Employee(2, "B", 56, 30000));
    empList.add(new Employee(3, "C", 42, 50000));
    empList.add(new Employee(4, "D", 52, 50000));
    empList.add(new Employee(5, "E", 32, 80000));
    empList.add(new Employee(6, "F", 72, 80000));

    return empList.stream();
  }
}

@Data
@AllArgsConstructor
class Employee
{
  private long id;
  private String name;
  private int age;
  private double salary;
}
```

Program output.

Output

```
false
```

3. Conclusion

`Stream.allMatch()` method can be useful in certain cases where we need to run a check on all stream elements.

For example, we can use `allMatch()` on a stream of `Employee` objects to validate if all employees are above a certain age.

It is **short-circuiting** operation. A terminal operation is short-circuiting if, when presented with infinite input, it may terminate in finite time.

Happy Learning !!

Sourcecode on Github

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. Java Stream reuse – traverse stream multiple times?
2. Java Stream count() Matches with filter()
3. Java Stream forEach()
4. Java Stream sorted()
5. Java Stream max()
6. Java Stream peek()
7. Java Stream limit()

8. Java Stream skip()

9. Java Stream findFirst()

0. Java Stream findAny()

2 thoughts on “Java Stream allMatch()”

Abhishek Prasad

July 17, 2021 at 5:34 pm

Example 2 needs to be evaluated, because the `.allMatch()` function will check for every employee and not for the employee who has age>50.

Reply

Lokesh Gupta

July 18, 2021 at 2:20 am

`allMatch()` will each **Employee** instance for both conditions.

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