TOP 10 JAVA 8 STREAMS PROGRAMS

JAVA INTERVIEW

01 Using Java 8, return a list containing only **even numbers** from a list of integers.

02 Using Java 8, find the **maximum value** in a list of integers.

03 Using Java 8, **sort a list** of integers in ascending order.

04 Using Java 8, **count** the elements in a list that are **greater than 5.**

05 Using Java 8, retrieve all **distinct elements** from a list.

06 Using Java 8, **skip the first 2 elements** of a list and return the rest.

07 Using Java 8, convert a list of integers to a Set to **remove duplicates.**

08 Using Java 8, **group elements** by a specific property, such as age.

09 Using Java 8, **reduce a list of integers** to their sum.

10 Using Java 8, **convert all strings** in a list to uppercase.

01. Using Java 8, return a list containing only even numbers from a list of integers.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.ArrayList;
import java.util.stream.Collectors;

// Way 1: Using Java 8 Streams to filter even numbers
public class EvenNumbersWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);

        List<Integer> evenNumbers = numbers.stream()
                                           .filter(n -> n % 2 == 0)
                                           .collect(Collectors.toList());

        System.out.println("Even Numbers (Stream): " + evenNumbers);
    }
}

// Way 2: Using a basic for-loop to filter even numbers
class EvenNumbersWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);
        List<Integer> evenNumbers = new ArrayList<>();

        for (Integer num : numbers) {
            if (num % 2 == 0) {
                evenNumbers.add(num);
            }
        }

        System.out.println("Even Numbers (Loop): " + evenNumbers);
    }
}
```

02. Using Java 8, find the maximum value in a list of integers.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.Optional;

// Way 1: Using Java 8 streams to find the maximum value
public class MaxValueWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 3, 7, 2, 9, 5);

        Optional<Integer> max = numbers.stream()
                                       .max(Integer::compare);

        System.out.println("Maximum Value (Stream): " + max.orElse(null));
    }
}

// Way 2: Using a basic for-loop to find the maximum value
class MaxValueWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 3, 7, 2, 9, 5);

        int max = Integer.MIN_VALUE;
        for (Integer num : numbers) {
            if (num > max) {
                max = num;
            }
        }

        System.out.println("Maximum Value (Loop): " + max);
    }
}
```

03. Using Java 8, sort a list of integers in ascending order.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

// Way 1: Using Java 8 streams to sort a list in ascending order
public class SortListWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(5, 1, 3, 7, 2);

        List<Integer> sortedNumbers = numbers.stream()
                                             .sorted() // Sort in natural (ascending) order
                                             .collect(Collectors.toList());

        System.out.println("Sorted List (Stream): " + sortedNumbers);
    }
}

import java.util.Collections;

// Way 2: Using Collections.sort() to sort a list in ascending order
class SortListWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(5, 1, 3, 7, 2);

        Collections.sort(numbers); // Sort using Collections.sort()

        System.out.println("Sorted List (Collections): " + numbers);
    }
}
```

04. Using Java 8, count the elements in a list that are greater than 5.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;

// Way 1: Using Java 8 streams to count elements greater than 5
public class CountGreaterThanFiveWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 3, 7, 2, 9, 5);

        long count = numbers.stream()
                            .filter(n -> n > 5)
                            .count();

        System.out.println("Input: " + numbers);
        System.out.println("Count of Numbers > 5: " + count);
    }
}

// Way 2: Using a basic for-loop to count elements greater than 5
class CountGreaterThanFiveWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 3, 7, 2, 9, 5);

        int count = 0;
        for (Integer num : numbers) {
            if (num > 5) {
                count++;
            }
        }

        System.out.println("Input: " + numbers);
        System.out.println("Count of Numbers > 5: " + count);
    }
}
```

05. Using Java 8, retrieve all distinct elements from a list.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

// Way 1: Using Java 8 streams to retrieve distinct elements
public class DistinctElementsWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 4, 5);

        List<Integer> distinctNumbers = numbers.stream()
                                               .distinct()
                                               .collect(Collectors.toList());

        System.out.println("Input: " + numbers);
        System.out.println("Distinct Elements: " + distinctNumbers);
    }
}

import java.util.ArrayList;

// Way 2: Using a basic loop to retrieve distinct elements
class DistinctElementsWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 4, 5);
        List<Integer> distinctNumbers = new ArrayList<>();

        for (Integer num : numbers) {
            if (!distinctNumbers.contains(num)) {
                distinctNumbers.add(num);
            }
        }

        System.out.println("Input: " + numbers);
        System.out.println("Distinct Elements: " + distinctNumbers);
    }
}
```

05. Using Java 8, retrieve all distinct elements from a list.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

// Way 1: Using Java 8 streams to retrieve distinct elements
public class DistinctElementsWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 4, 5);

        List<Integer> distinctNumbers = numbers.stream()
                                               .distinct()
                                               .collect(Collectors.toList());

        System.out.println("Input: " + numbers);
        System.out.println("Distinct Elements: " + distinctNumbers);
    }
}

import java.util.ArrayList;

// Way 2: Using a basic loop to retrieve distinct elements
class DistinctElementsWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 4, 5);
        List<Integer> distinctNumbers = new ArrayList<>();

        for (Integer num : numbers) {
            if (!distinctNumbers.contains(num)) {
                distinctNumbers.add(num);
            }
        }

        System.out.println("Input: " + numbers);
        System.out.println("Distinct Elements: " + distinctNumbers);
    }
}
```

06. Using Java 8, skip the first 2 elements of a list and return the rest.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */


import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

// Way 1: Using Java 8 streams to skip the first 2 elements
public class SkipElementsWay1
{
    public static void main(String[] args)
    {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);

        List<Integer> skippedNumbers = numbers.stream()
                                              .skip(2) // Skip first 2 elements
                                              .collect(Collectors.toList());

        System.out.println("Input: " + numbers);
        System.out.println("After Skipping First 2 Elements: " + skippedNumbers);
    }
}

// Way 2: Using subList to skip the first 2 elements
class SkipElementsWay2
{
    public static void main(String[] args)
    {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);

        List<Integer> skippedNumbers = numbers.subList(2, numbers.size()); // Skip first 2

        System.out.println("Input: " + numbers);
        System.out.println("After Skipping First 2 Elements: " + skippedNumbers);
    }
}
```

07. Using Java 8, convert a list of integers to a set to remove duplicates.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

// Way 1: Using Java 8 streams to remove duplicates by converting to a set
public class RemoveDuplicatesWay1
{
    public static void main(String[] args)
    {
        List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 4, 5);

        Set<Integer> uniqueNumbers = numbers.stream()
                                            .collect(Collectors.toSet());

        System.out.println("Input: " + numbers);
        System.out.println("Unique Numbers: " + uniqueNumbers);
    }
}

import java.util.HashSet;

// Way 2: Using a HashSet to remove duplicates from a list
class RemoveDuplicatesWay2
{
    public static void main(String[] args)
    {
        List<Integer> numbers = Arrays.asList(1, 2, 2, 3, 4, 4, 5);

        Set<Integer> uniqueNumbers = new HashSet<>(numbers); // HashSet removes duplicates

        System.out.println("Input: " + numbers);
        System.out.println("Unique Numbers: " + uniqueNumbers);
    }
}
```

08. Using Java 8, group elements by a specific property, such as age.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.*;
import java.util.stream.Collectors;

// Simple Person class with name and age
class Person {
    String name;
    int age;

    // Constructor
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getter for age
    public int getAge() {
        return age;
    }

    public String toString() {
        return name + " (" + age + ")";
    }
}
```

```java
// Way 1: Using Java 8 streams to group people by age
public class GroupByPropertyStream {
    public static void main(String[] args) {
        List<Person> people = Arrays.asList(
            new Person("Dhoni", 41),
            new Person("Sachin", 50),
            new Person("Kohli", 34),
            new Person("Rohit", 34),
            new Person("Pant", 25)
        );

        Map<Integer, List<Person>> groupedByAge = people.stream()
            .collect(Collectors.groupingBy(Person::getAge));

        System.out.println("Grouped By Age (Java 8): " + groupedByAge);
    }
}

// Way 2: Using a basic loop to group people by age
class GroupByPropertyLoop {
    public static void main(String[] args) {
        List<Person> people = Arrays.asList(
            new Person("Dhoni", 41),
            new Person("Sachin", 50),
            new Person("Kohli", 34),
            new Person("Rohit", 34),
            new Person("Pant", 25)
        );

        Map<Integer, List<Person>> groupedByAge = new HashMap<>();
        for (Person person : people) {
            groupedByAge.putIfAbsent(person.getAge(), new ArrayList<>());
            groupedByAge.get(person.getAge()).add(person);
        }

        System.out.println("Grouped By Age (Basic Loop): " + groupedByAge);
    }
}
```

09. Using Java 8, reduce a list of integers to their sum.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;

// Way 1: Using Java 8 streams to calculate the sum of integers
public class SumOfListWay1 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

        int sum = numbers.stream()
                         .reduce(0, Integer::sum);

        System.out.println("Input: " + numbers);
        System.out.println("Sum: " + sum);
    }
}

// Way 2: Using a basic loop to calculate the sum of integers
class SumOfListWay2 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

        int sum = 0;
        for (Integer num : numbers) {
            sum += num;
        }

        System.out.println("Input: " + numbers);
        System.out.println("Sum: " + sum);
    }
}
```

10. Using Java 8, convert all strings in a list to uppercase.

```java
/*
 * Author    : @dev_roadmaps (Instagram)
 * Website   : https://devroadmaps.in
 * Copyright : © 2025 Dev Roadmaps. All rights reserved.
 */

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

// Way 1: Using Java 8 streams to convert strings to uppercase
public class ConvertToUpperCaseWay1 {
    public static void main(String[] args) {
        List<String> strings = Arrays.asList("java", "stream", "api");

        List<String> upperCaseStrings = strings.stream()
                                               .map(String::toUpperCase)
                                               .collect(Collectors.toList());

        System.out.println("Input: " + strings);
        System.out.println("Uppercase Strings: " + upperCaseStrings);
    }
}

import java.util.ArrayList;

// Way 2: Using a basic loop to convert strings to uppercase
class ConvertToUpperCaseWay2 {
    public static void main(String[] args) {
        List<String> strings = Arrays.asList("java", "stream", "api");
        List<String> upperCaseStrings = new ArrayList<>();

        for (String str : strings) {
            upperCaseStrings.add(str.toUpperCase());
        }

        System.out.println("Input: " + strings);
        System.out.println("Uppercase Strings: " + upperCaseStrings);
    }
}
```

📲 Join the Dev Roadmaps Community

In just **6 months**, we've built a strong learning network:

✅ **70K+ followers on Instagram**
💬 **8,600+ members on WhatsApp**
💼 **Thousands following us on LinkedIn** for job updates and referrals

👉 **Follow us on Instagram: @dev_roadmaps**
💬 **Join our WhatsApp channel** for exclusive Java content
💼 **Follow us on LinkedIn** for referrals and job openings

Let's keep growing, learning, and winning — **together** 💪