**Contoh Streaming di Java:**

Berikut ini contoh yang menunjukkan penggunaan Streams di Java untuk memfilter dan mengubah kumpulan nama:

List<String> names = Arrays.asList("John", "Jane", "Adam", "Eve", "Mike");

List<String> filteredNames = names.stream()

.filter(name -> name.startsWith("J"))

.map(String::toUpperCase)

.collect(Collectors.toList());

System.out.println(filteredNames); // Output: [JOHN, JANE]

Dalam contoh ini, kita membuat aliran dari daftar nama, memfilter nama yang dimulai dengan "J", mengubahnya menjadi huruf besar menggunakan operasi peta, dan mengumpulkan hasilnya ke dalam daftar baru menggunakan operasi terminal kumpulkan. Outputnya adalah nama yang difilter dan diubah: [JOHN, JANE].

## **Operasi Berbeda pada Aliran**

Stream menyediakan berbagai operasi yang dapat dirangkai bersama untuk membuahkan hasil. Operasi aliran dapat diklasifikasikan menjadi dua jenis.

* Operasi Menengah
* Operasi Terminal

### 1. Operasi Menengah

Operasi perantara mengembalikan aliran sebagai keluaran, dan operasi perantara tidak dijalankan sampai operasi terminal dipanggil pada aliran. Ini disebut evaluasi malas, dan dibahas secara rinci di bagian selanjutnya (Evaluasi Malas).

**Saring()**

Metode filter() mengembalikan aliran dengan elemen aliran yang cocok dengan predikat yang diberikan. Predikat adalah antarmuka fungsional di Java yang menerima satu masukan dan dapat mengembalikan nilai boolean.

**Contoh**

**public class Main {**

**public static void main(String[] args) {**

**final List<Integer> list = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));**

**final List<Integer> ans = list.stream()**

**.filter(value -> value % 2 == 0)**

**.collect(Collectors.toList());**

**System.out.println(Arrays.toString(ans.toArray()));**

**}**

**}**

**Keluaran:** **[2, 4]**

**peta()**

[Metode map ()](https://www.scaler.com/topics/java-stream-map/) mengembalikan aliran dengan elemen yang dihasilkan setelah menerapkan fungsi yang diberikan pada elemen aliran.

**Contoh**

public class Main {

public static void main(String[] args) {

final List<Integer> list = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

final List<Integer> ans = list.stream()

.map(value -> value \* 10)

.collect(Collectors.toList());

System.out.println(Arrays.toString(ans.toArray()));

}

}

Keluaran: [10, 20, 30, 40, 50]

**diurutkan()**

Metode sortir() mengembalikan aliran dengan elemen aliran yang diurutkan berdasarkan tatanan alami atau Pembanding yang disediakan.

**Contoh**

public class Main {

public static void main(String[] args) {

final List<Integer> list = new ArrayList<>(Arrays.asList(5, 1, 3, 4, 2));

System.out.println("Ascending Order");

list.stream().sorted()

.forEach(System.out::println);

System.out.println("\nDescending Order");

list.stream().sorted(Comparator.reverseOrder())

.forEach(System.out::println);

}

}

Keluaran: Ascending Order

1

2

3

4

5

Descending Order

5

4

3

2

1

**berbeda()**

Metode berbeda() ini mengembalikan aliran yang terdiri dari elemen aliran yang berbeda (yaitu) metode ini menghapus elemen duplikat.

**Contoh**

public class Main {

public static void main(String[] args) {

final List<Integer> list = new ArrayList<>(Arrays.asList(1, 1, 2, 2, 3));

final List<Integer> ans = list.stream()

.distinct()

.collect(Collectors.toList());

System.out.println("Distinct List: " + Arrays.toString(ans.toArray()));

}

}

Keluaran: Distinct List: [1, 2, 3]

**mengintip()**

Metode peek() mengembalikan aliran yang terdiri dari elemen aliran setelah melakukan tindakan yang disediakan pada setiap elemen. Ini berguna ketika kita ingin mencetak nilai setelah setiap operasi perantara.

**Contoh**

public class Main {

public static void main(String[] args) {

final List<Integer> list = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

final List<Integer> ans = list.stream()

.filter(value -> value % 2 == 0)

.peek(value -> System.out.println("Filtered " + value))

.map(value -> value \* 10)

.collect(Collectors.toList());

System.out.println(Arrays.toString(ans.toArray()));

}

}

Keluaran: Filtered 2

Filtered 4

[20, 40]