Nama : Ahmad Zulveron
NOBP : 2111083003
Kelas : TRPL 3A

Java Streams

Java Streams adalah salah satu fitur yang kuat yang diperkenalkan dalam Java 8 untuk memanipulasi dan mengoperasikan koleksi data dengan cara yang deklaratif dan fungsional.

Apa Itu Java Streams?

Java Streams adalah alat untuk memanipulasi dan mengolah data dalam koleksi Java (seperti List, Set, Map, dll.) secara efisien dan deklaratif. Stream adalah urutan elemen yang dapat Anda operasikan dengan berbagai operasi seperti filter, map, reduce, dan banyak lagi. Dengan Java Streams, Anda dapat menghindari penggunaan loop for tradisional dan menggantinya dengan kode yang lebih ringkas dan ekspresif.

Cara Menggunakan Java Streams

Untuk menggunakan Java Streams, Anda perlu mengimpor paket java.util.stream. Di bawah ini adalah langkah-langkah dasar dalam mengoperasikan koleksi menggunakan Streams:

1. **Mengubah Koleksi Menjadi Stream:** Anda dapat mengubah koleksi menjadi Stream menggunakan metode stream() pada koleksi tersebut. Contoh:

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
Stream<Integer> stream = numbers.stream();
```

2. **Operasi pada Stream:** Anda dapat melakukan berbagai operasi pada Stream. Beberapa operasi umum adalah filter, map, reduce, collect, dan banyak lagi.

3. **Mengakhiri Stream:** Setelah operasi selesai, Anda biasanya akan mengakhiri Stream dengan operasi terminal seperti forEach, collect, reduce, count, dan lainnya.

4. **Parallel Streams:** Java juga mendukung Stream paralel dengan metode parallelStream(). Ini memungkinkan Anda untuk memproses elemen Stream secara paralel, yang dapat meningkatkan kinerja pada koleksi besar.

Contoh Program

Contoh 1: Menjumlahkan Bilangan Ganjil

Program ini mengambil sebuah list (daftar) bilangan bulat dan menghitung jumlah semua bilangan ganjil dalam list tersebut menggunakan Java Streams.

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

public class StreamExample {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

        int sumOfOddNumbers = numbers.stream()
            .filter(n -> n % 2 != 0)
            .mapToInt(Integer::intValue)
            .sum();

        System.out.println("Jumlah bilangan ganjil: " + sumOfOddNumbers);
    }
}
```

- Program ini membuat list numbers yang berisi bilangan bulat.
- Menggunakan Java Streams, program ini melakukan filter pada bilangan ganjil dengan menggunakan metode filter.
- Kemudian, program ini menggunakan mapToInt untuk mengonversi Stream menjadi IntStream, yang memungkinkan kita untuk menggunakan metode sum.
- Akhirnya, program menampilkan hasil jumlah bilangan ganjil.

Contoh 2: Mengalikan Semua Elemen dengan 2

Program ini mengambil list bilangan bulat dan mengalikan semua elemennya dengan 2 menggunakan Java Streams.

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

public class StreamExample {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

        List<Integer> doubledNumbers = numbers.stream()
            .map(n -> n * 2)
            .collect(Collectors.toList());

        System.out.println("Hasil perkalian: " + doubledNumbers);
    }
}
```

- Program ini membuat list numbers yang berisi bilangan bulat.
- Menggunakan Java Streams, program ini menggunakan metode map untuk mengalikan setiap elemen dengan 2.
- Hasilnya dikumpulkan kembali menjadi List menggunakan metode collect.
- Akhirnya, program menampilkan hasil perkalian.

Contoh 3: Menggunakan Parallel Stream

Program ini menggunakan Parallel Stream untuk menghitung jumlah bilangan genap dalam sebuah list.

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

public class ParallelStreamExample {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

        long count = numbers.parallelStream()
            .filter(n -> n % 2 == 0)
            .count();

        System.out.println("Jumlah bilangan genap: " + count);
    }
}
```

- Program ini membuat list numbers yang berisi bilangan bulat.
- Menggunakan Parallel Stream dengan metode parallelStream, program ini melakukan filter pada bilangan genap dengan menggunakan metode filter.
- Menggunakan metode count untuk menghitung jumlah bilangan genap.
- Akhirnya, program menampilkan hasil jumlah bilangan genap.

Semua contoh program di atas menggunakan Java Streams untuk mengoperasikan data koleksi dengan cara yang lebih deklaratif dan fungsional. Dengan Java Streams, Anda dapat melakukan berbagai operasi pada data dengan kode yang lebih ringkas dan mudah dibaca.