forEach() method

👉 forEach() in Stream API:

The forEach() method in Java's Stream API is used to iterate over elements in a stream and perform an action on each element.

👉 Key Points:

- **Introduced in Java 8** with the Stream API
- **Purpose**: Iterates through elements and performs an action on each one
- **Takes a Consumer<T>** functional interface as its parameter
- **Processes elements sequentially**, one at a time
- **Consumer<T>** is from the java.util.function package
  - It has a single method: accept(T t)
  - This method performs the operation on each element
- **Best used for**: Side effects like printing, logging, or updating external variables

👉 Example 1: Using Consumer Interface Explicitly

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

public class Demo {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(4, 5, 7, 3, 2, 6);

        // Creating a Consumer implementation
        Consumer<Integer> con = (Integer n) -> {
            System.out.println(n);
        };

        // Using forEach with the Consumer
        nums.forEach(con);
    }
}
```

Output:

```
4
5
7
3
2
6
```

👉 Example 2: Using Lambda Expression Directly

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

public class Demo {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(4, 5, 7, 3, 2, 6);

        // Using forEach with inline lambda
        nums.forEach(n -> System.out.println(n));
    }
}
```

Output:

```
4
5
7
3
2
6
```

👉 Remember:

- forEach() is efficient for iterating over stream elements.
- Ideal for logging, debugging, and performing non-transformational operations.