TELUSKO

Method Reference

👉 Method Reference

A method reference is a shorthand notation for a lambda expression that executes just one method. Instead of writing the complete lambda expression with parameters, you can simply refer to an existing method by its name using the **::** (double colon) operator.

👉 Key Points

- Method references use the syntax ClassName::methodName or objectName::methodName
- They provide a more concise way to write lambda expressions that only call a single method
- The **::** (double colon) operator is used to separate the class or object from the method name
- Method references work only when the lambda expression is simply calling another method
- They make your code shorter and often more readable

👉 Method Reference Example

- Let's start with a simple list of names that we'll work with:

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

public class MethodRefEx {
    public static void main(String[] args) {
        // Create a list of names
        List<String> names = Arrays.asList("Navin", "Harsh", "John");
        System.out.println("Original names: " + names);
    }
}
```

Output:

```
Original names: [Navin, Harsh, John]
```

TELUSKO

- Now, let's use the Stream API with a lambda expression to convert all names to uppercase:

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

public class MethodRefEx {
    public static void main(String[] args) {
        // Create a list of names
        List<String> names = Arrays.asList("Navin", "Harsh", "John");
        System.out.println("Original names: " + names);

        // Convert to uppercase using lambda expression
        List<String> uNames = names.stream()
                                   .map(name -> name.toUpperCase())
                                   .toList();

        System.out.println("Uppercase names (using lambda): " + uNames);
    }
}
```

Output:

```
Original names: [Navin, Harsh, John]
Uppercase names (using lambda): [NAVIN, HARSH, JOHN]
```

- We can simplify the code above by using a method reference:

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

public class MethodRefEx {
    public static void main(String[] args) {
        // Create a list of names
        List<String> names = Arrays.asList("Navin", "Harsh", "John");
        System.out.println("Original names: " + names);

        // Convert to uppercase using method reference
        List<String> uNames = names.stream()
                                   .map(String::toUpperCase)
                                   .toList();

        System.out.println("Uppercase names (using method reference): " + uNames);

        // Using method reference with forEach
        System.out.print("Printing each name: ");
        uNames.forEach(System.out::println);
    }
}
```

Output:

```
Original names: [Navin, Harsh, John]
Uppercase names (using method reference): [NAVIN, HARSH, JOHN]
Printing each name: NAVIN
HARSH
JOHN
```

Explanation:

- name -> name.toUpperCase() becomes String::toUpperCase
  - Here, we're saying "call the toUpperCase method on the String object you receive"
- name -> System.out.println(name) becomes System.out::println
  - Here, we're saying "call the println method of System.out with the object you receive"

This makes our code shorter and often easier to read, especially once you get familiar with the syntax.

👉 Types of Method References

1. **Static method reference**: ClassName::staticMethodName
2. **Instance method reference of a particular object**:
   objectName::instanceMethodName
3. **Instance method reference of an arbitrary object of a particular type**:
   ClassName::instanceMethodName
4. **Constructor reference**: ClassName::new

TELUSKO