Using Method References to Write Lambda Expressions

José Paumard

PhD, Java Champion, JavaOne Rockstar

@JosePaumard | https://github.com/JosePaumard

Making your code even cleaner

Module Overview

What is a method reference?

What are the four types of method reference?

How to avoid traps!

Method references in action

Writing Lambdas in Another Way

```
var ints = List.of(1, 2, 3);
ints.forEach(System.out::println);
```

```
var ints = List.of(1, 2, 3);
Consumer<Integer> println =
   System.out::println
ints.forEach(println);
```

```
var ints = List.of(1, 2, 3);
Consumer<Integer> println =
    i -> System.out.println(i);
ints.forEach(println);
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;

Predicate<String> isEmpty =
   s -> s.isEmpty();
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;

Predicate<String> isEmpty =
   s -> s.isEmpty();
Predicate<String> isEmptyMR =
   String::isEmpty;
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;

Predicate<String> isEmpty =
   s -> s.isEmpty();
Predicate<String> isEmptyMR =
   String::isEmpty;

BiFunction<String, String, Integer> indexOf =
   (word, sentence) -> sentence.indexOf(word);
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;

Predicate<String> isEmpty =
   s -> s.isEmpty();
Predicate<String> isEmptyMR =
   String::isEmpty;

BiFunction<String, String, Integer> indexOf =
   (word, sentence) -> sentence.indexOf(word);
BiFunction<String, String, Integer> lengthMR =
   word::indexOf;
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;

Predicate<String> isEmpty =
   s -> s.isEmpty();
Predicate<String> isEmptyMR =
   String::isEmpty;

BiFunction<String, String, Integer> indexOf =
   (word, sentence) -> sentence.indexOf(word);
BiFunction<String, String, Integer> lengthMR =
   word::indexOf;

UnaryOperator<Double> sqrt =
   d -> Math.sqrt(d);
```

```
Supplier<List<String>> build =
   () -> new ArrayList<>();
Supplier<List<String>> buildMR =
   ArrayList::new;

Consumer<String> print =
   s -> System.out.println(s);
Consumer<String> printMR =
   System.out::println;

Predicate<String> isEmpty =
   s -> s.isEmpty();
Predicate<String> isEmptyMR =
   String::isEmpty;

BiFunction<String, String, Integer> indexOf =
   (word, sentence) -> sentence.indexOf(word);
BiFunction<String, String, Integer> lengthMR =
   word::indexOf;

UnaryOperator<Double> sqrt =
   d -> Math.sqrt(d);
UnaryOperator<Double> sqrtMR =
   Math::sqrt;
```

The Four Types of Method References

Constructor Method References

```
// Constructor method reference syntax: Type::new

Supplier<List<String>> sup   = () -> new ArrayList<>();
Supplier<List<String>> supMR = ArrayList::new;
```

Constructor Method References

```
// Constructor method reference syntax: Type::new

Supplier<List<String>> sup   = () -> new ArrayList<>();
Supplier<List<String>> supMR = ArrayList::new;

Function<Integer, List<String>> function   = size -> new ArrayList<>(size);
Function<Integer, List<String>> functionMR = ArrayList::new;
```

Constructor Method References

```
// Constructor method reference syntax: Type::new

Supplier<List<String>> sup   = () -> new ArrayList<>();
Supplier<List<String>> supMR = ArrayList::new;

Function<Integer, List<String>> function   = size -> new ArrayList<>(size);
Function<Integer, List<String>> functionMR = ArrayList::new;

IntFunction<String[]> arrayNew   = length -> new String[length];
IntFunction<String[]> arrayNewMR = String[]::new;
```

Static Method References

```
// Static method reference syntax: Type::staticMethod

Supplier<List<String>> sup   = () -> Collections.emptyList();
Supplier<List<String>> supMR = Collections::emptyList;
```

Static Method References

```
// Static method reference syntax: Type::staticMethod

Supplier<List<String>> sup   = () -> Collections.emptyList();
Supplier<List<String>> supMR = Collections::emptyList;

DoubleUnaryOperator sqrt   = d -> Math.sqrt(d);
DoubleUnaryOperator sqrtMR = Math::sqrt;
```

Static Method References

```
// Static method reference syntax: Type::staticMethod

Supplier<List<String>> sup   = () -> Collections.emptyList();
Supplier<List<String>> supMR = Collections::emptyList;

DoubleUnaryOperator sqrt   = d -> Math.sqrt(d);
DoubleUnaryOperator sqrtMR = Math::sqrt;

DoublePredicate valueOf   = d -> Double.isFinite(d);
DoublePredicate valueOfMR = Double::isFinite;
```

Static Method References

```
// Static method reference syntax: Type::staticMethod

Supplier<List<String>> sup   = () -> Collections.emptyList();
Supplier<List<String>> supMR = Collections::emptyList;

DoubleUnaryOperator sqrt   = d -> Math.sqrt(d);
DoubleUnaryOperator sqrtMR = Math::sqrt;

DoublePredicate valueOf   = d -> Double.isFinite(d);
DoublePredicate valueOfMR = Double::isFinite;

IntBinaryOperator max   = (i1, i2) -> Integer.max(i1, i2);
IntBinaryOperator maxMR = Integer::max;
```

Bound and Unbound Method References

```
// Bound method reference syntax: expression::instanceMethod
```

```
// Unbound method reference syntax: Type::instanceMethod
```

Bound and Unbound Method References

```
// Bound method reference syntax: expression::instanceMethod

Consumer<String> printer   = s -> System.out.println(s);
Consumer<String> printerMR = System.out::println;

// Unbound method reference syntax: Type::instanceMethod
```

Bound and Unbound Method References

```
// Bound method reference syntax: expression::instanceMethod

Consumer<String> printer   = s -> System.out.println(s);
Consumer<String> printerMR = System.out::println;

// Unbound method reference syntax: Type::instanceMethod

Predicate<String> isEmpty   = s -> s.isEmpty();
Predicate<String> isEmptyMR = String::isEmpty;
```

Bound Method References

```
// Bound method reference syntax: expression::instanceMethod

Consumer<String> printer   = s -> System.out.println(s);
Consumer<String> printerMR = System.out::println;
```

Bound Method References

```
// Bound method reference syntax: expression::instanceMethod

Consumer<String> printer   = s -> System.out.println(s);
Consumer<String> printerMR = System.out::println;

ToIntFunction<String> indexOf   = s -> "hello world".indexOf(s);
ToIntFunction<String> indexOfMR = "hello world"::indexOf;
```

Bound Method References

```
// Bound method reference syntax: expression::instanceMethod

Consumer<String> printer   = s -> System.out.println(s);
Consumer<String> printerMR = System.out::println;

ToIntFunction<String> indexOf   = s -> "hello world".indexOf(s);
ToIntFunction<String> indexOfMR = "hello world"::indexOf;

ToIntBiFunction<String, Integer> indexOfFromIndex   =
   (s, index) -> "hello world".indexOf(s, index);
ToIntBiFunction<String, Integer> indexOfFromIndexMR =
   "hello world"::indexOf;
```

Unbound Method References

```
// Unbound method reference syntax: Type::instanceMethod

Predicate<String> isEmpty   = s -> s.isEmpty();
Predicate<String> isEmptyMR = String::isEmpty;
```

Unbound Method References

```
// Unbound method reference syntax: Type::instanceMethod

Predicate<String> isEmpty   = s -> s.isEmpty();
Predicate<String> isEmptyMR = String::isEmpty;

ToIntBiFunction<String, String> indexOf2 =
   (sentence, word) -> sentence.indexOf(word);
ToIntBiFunction<String, String> indexOf2MR =
   String::indexOf;
```

Unbound Method References

```
// Unbound method reference syntax: Type::instanceMethod

Predicate<String> isEmpty   = s -> s.isEmpty();
Predicate<String> isEmptyMR = String::isEmpty;

ToIntBiFunction<String, String> indexOf2 =
   (sentence, word) -> sentence.indexOf(word);
ToIntBiFunction<String, String> indexOf2MR =
   String::indexOf;

ToIntBiFunction<String, String> indexOf3 =
   (word, sentence) -> sentence.indexOf(word);
```

Unbound Method References

```
// Unbound method reference syntax: Type::instanceMethod
```

```
ToIntTriFunction<String, String, Integer> indexOfFromIndex2   =
   (sentence, word, index) -> sentence.indexOf(word, index);
ToIntTriFunction<String, String, Integer> indexOfFromIndex2MR =
   String::indexOf;
```

Unbound Method References

```
// Unbound method reference syntax: Type::instanceMethod

@FunctionalInterface
interface ToIntTriFunction<T, U, V> {
    int apply(T t, U u, V v);
}


ToIntTriFunction<String, String, Integer> indexOfFromIndex2   =
   (sentence, word, index) -> sentence.indexOf(word, index);
ToIntTriFunction<String, String, Integer> indexOfFromIndex2MR =
   String::indexOf;
```

Method References in Action

Method Rerences in Action

Writing simple method references

Avoiding traps!

Making your code cleaner

Module Summary

What is a method reference?

How you can write them

How to understand how they are working

How they can make your code cleaner

Up Next:

Creating Comparators Using Method References