Java 8 Method References

Douglas C. Schmidt
Learning Objectives in This Lesson

- Recognize foundational functional programming features in Java 8, e.g.,
  - Lambda expressions
  - Method (and constructor) references

Several concise examples are used to showcase foundational Java 8 features.
Overview of Method References
Overview of Method References

A method reference is a compact, easy-to-read handle for a method that already has a name.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Syntax</th>
<th>Method Reference</th>
<th>Lambda Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reference to a static method</td>
<td>ContainingClass::staticMethodName</td>
<td>String::valueOf</td>
<td>s -&gt; String.valueOf(s)</td>
</tr>
<tr>
<td>2. Reference to an instance method of a particular object</td>
<td>containingObject::instanceMethodName</td>
<td>s::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>3. Reference to instance method of an arbitrary object of a given type</td>
<td>ContainingType::methodName</td>
<td>String::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>4. Reference to a constructor</td>
<td>ClassName::new</td>
<td>String::new</td>
<td>() -&gt; new String()</td>
</tr>
</tbody>
</table>

See [docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html](docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html)
Overview of Method References

A method reference is a compact, easy-to-read handle for a method that already has a name.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Syntax</th>
<th>Method Reference</th>
<th>Lambda Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reference to a static method</td>
<td>ContainingClass::staticMethodName</td>
<td>String::valueOf</td>
<td>s -&gt; String.valueOf(s)</td>
</tr>
<tr>
<td>2. Reference to an instance method of a particular object</td>
<td>containingObject::instanceMethodName</td>
<td>s::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>3. Reference to instance method of an arbitrary object of a given type</td>
<td>ContainingType::methodName</td>
<td>String::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>4. Reference to a constructor</td>
<td>ClassName::new</td>
<td>String::new</td>
<td>() -&gt; new String()</td>
</tr>
</tbody>
</table>

It’s shorthand syntax for a lambda expression that executes one method.
Overview of Method References

- A method reference is a compact, easy-to-read handle for a method that already has a name.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Syntax</th>
<th>Method Reference</th>
<th>Lambda Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reference to a static method</td>
<td>ContainingClass::staticMethodName</td>
<td>String::valueOf</td>
<td>s -&gt; String.valueOf(s)</td>
</tr>
<tr>
<td>2. Reference to an instance method of a particular object</td>
<td>containingObject::instanceMethodName</td>
<td>s::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>3. Reference to instance method of an arbitrary object of a given type</td>
<td>ContainingType::methodName</td>
<td>String::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>4. Reference to a constructor</td>
<td>ClassName::new</td>
<td>String::new</td>
<td>() -&gt; new String()</td>
</tr>
</tbody>
</table>
### Overview of Method References

- A method reference is a compact, easy-to-read handle for a method that already has a name.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Syntax</th>
<th>Method Reference</th>
<th>Lambda Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reference to a static method</td>
<td>ContainingClass::staticMethodName</td>
<td>String::valueOf</td>
<td>s -&gt; String.valueOf(s)</td>
</tr>
<tr>
<td>2. Reference to an instance method of a particular object</td>
<td>containingObject::instanceMethodName</td>
<td>s::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>3. Reference to instance method of an arbitrary object of a given type</td>
<td>ContainingType::methodName</td>
<td>String::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>4. Reference to a constructor</td>
<td>ClassName::new</td>
<td>String::new</td>
<td>() -&gt; new String()</td>
</tr>
</tbody>
</table>
### Overview of Method References

- A method reference is a compact, easy-to-read handle for a method that already has a name.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Syntax</th>
<th>Method Reference</th>
<th>Lambda Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reference to a static method</td>
<td><code>ContainingClass::staticMethodName</code></td>
<td><code>String::valueOf</code></td>
<td><code>s -&gt; String.valueOf(s)</code></td>
</tr>
<tr>
<td>2. Reference to an instance method of a particular object</td>
<td><code>containingObject::instanceMethodName</code></td>
<td><code>s::toString</code></td>
<td><code>s -&gt; s.toString()</code></td>
</tr>
<tr>
<td>3. Reference to instance method of an arbitrary object of a given type</td>
<td><code>ContainingType::methodName</code></td>
<td><code>String::toString</code></td>
<td><code>s -&gt; s.toString()</code></td>
</tr>
<tr>
<td>4. Reference to a constructor</td>
<td><code>ClassName::new</code></td>
<td><code>String::new</code></td>
<td><code>() -&gt; new String()</code></td>
</tr>
</tbody>
</table>
A method reference is a compact, easy-to-read handle for a method that already has a name.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Syntax</th>
<th>Method Reference</th>
<th>Lambda Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Reference to a static method</td>
<td>ContainingClass::staticMethodName</td>
<td>String::valueOf</td>
<td>s -&gt; String.valueOf(s)</td>
</tr>
<tr>
<td>2. Reference to an instance method of a particular object</td>
<td>containingObject::instanceMethodName</td>
<td>s::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>3. Reference to instance method of an arbitrary object of a given type</td>
<td>ContainingType::methodName</td>
<td>String::toString</td>
<td>s -&gt; s.toString()</td>
</tr>
<tr>
<td>4. Reference to a constructor</td>
<td>ClassName::new</td>
<td>String::new</td>
<td>() -&gt; new String()</td>
</tr>
</tbody>
</table>
Overview of Method References

- Method references are more compact than alternative mechanisms.

Java 8 method references support even more concise “behavioral parameterization”

See blog.indrek.io/articles/java-8-behavior-parameterization
Overview of Method References

- Methods references are more compact than lambdas & anonymous classes.

```

Arrays.sort(nameArray, new Comparator<String>(){
    public int compare(String s, String t) { return s.toLowerCase().compareTo(t.toLowerCase()); }});
```

vs.

```
Arrays.sort(nameArray,
    (s, t) -> s.compareToIgnoreCase(t));
```

vs. **Method references are even more compact & readable.**

```
Arrays.sort(nameArray, String::compareToIgnoreCase);
```

See [www.gravytrain.co.uk/blog/java-8-an-introduction-to-method-references](http://www.gravytrain.co.uk/blog/java-8-an-introduction-to-method-references)
Methods references are more compact than lambdas & anonymous classes.

```java

Arrays.sort(nameArray, new Comparator<String>(){
    public int compare(String s, String t) { return s.toLowerCase().compareTo(t.toLowerCase()); }
});

VS.

Arrays.sort(nameArray, (s, t) -> s.compareToIgnoreCase(t));

VS. Method references also promote code reuse.

Arrays.sort(nameArray, String::compareToIgnoreCase);
```

The Arrays.sort() implementation doesn’t change, but the params do!
Methods references are more compact than lambdas & anonymous classes.


Arrays.sort(nameArray, new Comparator<String>(){
   public int compare(String s, String t) { return s.toLowerCase().compareTo(t.toLowerCase()); }});

VS.

Arrays.sort(nameArray, (s, t) -> s.compareToIgnoreCase(t));

VS. Replacing one comparison with another is easy, a la the Strategy pattern.

Arrays.sort(nameArray, String::compareTo);

See en.wikipedia.org/wiki/Strategy_pattern
Overview of Method References

- Methods references are more compact than lambdas & anonymous classes.

```java
String[] nameArray = {"Barbara", "James", "Mary", "John", 
"Robert", "Michael", "Linda", "james", "mary"};

Arrays.sort(nameArray, new Comparator<String>(){
    public int compare(String s, String t) { return 
        s.toLowerCase().compareTo(t.toLowerCase()); }})
```

VS.

```java
Arrays.sort(nameArray,
    (s, t) -> s.compareToIgnoreCase(t));
```

VS.

```java
Arrays.sort(nameArray, String::compareTo);
```

Therefore, it’s best practice to use method references whenever you can!
Douglas C. Schmidt

Applying Method References in Practice
Method references can be used to print a collection or array in various ways.

```java
```

*Array of names represented as strings.*
Applying Method References in Practice

- Method references can be used to print a collection or array in various ways.
  ```java
  ```

- `System.out.println()` can be used to print out an array
  ```java
  System.out.println(Arrays.asList(nameArray));
  ```
  prints:
  ```java
  [Barbara, James, Mary, John, Linda, Michael, Linda, james, mary]
  ```

See docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html#println
Method references can be used to print a collection or array in various ways.

```java
```

System.out.println() can be used to print out an array

```java
System.out.println(Arrays.asList(nameArray));
```

prints:

```
[Barbara, James, Mary, John, Linda, Michael, Linda, James, Mary]
```

See [www.tutorialspoint.com/java/util/arrays_aslist.htm](http://www.tutorialspoint.com/java/util/arrays_aslist.htm)
Applying Method References in Practice

- Method references can be used to print a collection or array in various ways.
  ```java
  ```

  - `System.out.println()` can be used to print out an array.
  - Java 8’s `forEach()` method can be used to print out values of an array.

Method references can be used to print a collection or array in various ways.

```java
```

- System.out.println() can be used to print out an array.
- Java 8’s forEach() method can be used to print out values of an array, e.g.,

  ```java
  Stream.of(nameArray).forEach(System.out::print);
  ```

  prints:

  ```
  Factory method that creates a stream from an array.
  BarbaraJamesMaryJohnLindaMichaelLindajamesmary
  ```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of)
Applying Method References in Practice

- Method references can be used to print a collection or array in various ways.

```java
```

- `System.out.println()` can be used to print out an array.
- Java 8’s `forEach()` method can be used to print out values of an array, e.g.,
  - In conjunction with a stream & method reference
    ```java
    Stream.of(nameArray).forEach(System.out::print);
    ```
    prints:
    ```
    BarbaraJamesMaryJohnLindaMichaelLindajamesmary
    ```

Performing method reference action on each stream element.

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#forEach
Method references can be used to print a collection or array in various ways.

```java
String[] nameArray = {
  "Barbara", "James", "Mary", "John",
  "Robert", "Michael", "Linda", "james", "mary"};
```

- `System.out.println()` can be used to print out an array
- Java 8’s `forEach()` method can be used to print out values of an array, e.g.,
  - In conjunction with a stream & method reference
  ```java
  Arrays.asList(nameArray).forEach(System.out::println);
  ```
  - In conjunction with a collection (e.g., List)
  ```java
  Arrays.asList(nameArray).forEach(System.out::println);
  ```
  prints:

```
BarbaraJamesMaryJohnLindaMichaelLindajamesmary
```

See [docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#asList](https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#asList)
Applying Method References in Practice

- Method references can be used to print a collection or array in various ways.
  ```java
  ```
- System.out.println() can be used to print out an array.
- Java 8’s forEach() method can be used to print out values of an array, e.g.,
  ```java
  Arrays.asList(nameArray).forEach(System.out::print);
  ```
  prints:
  
  Performs method reference action on each list element.
  
  BarbaraJamesMaryJohnLindaMichaelLindajamesmary

See docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html#forEach
Applying Method References in Practice

• Method references can be used to print a collection or array in various ways.
  ```java
  ```

• System.out.println() can be used to print out an array

• Java 8’s forEach() method can be used to print out values of an array, e.g.,
  ```java
  • In conjunction with a stream & method reference
  • In conjunction with a collection (e.g., List)
  • forEach() on a stream differs slightly from forEach() on a collection
  ```

See stackoverflow.com/a/23232560
Applying Method References in Practice

- Method references can be used to print a collection or array in various ways.
  
  ```java
  ```

- System.out.println() can be used to print out an array.
- Java 8’s forEach() method can be used to print out values of an array, e.g.,
  - In conjunction with a stream & method reference
  - In conjunction with a collection (e.g., List)
  - forEach() on a stream differs slightly from forEach() on a collection
    - e.g., forEach() ordering is undefined on a stream, whereas it’s defined for a collection.

See stackoverflow.com/a/23232560