Overview of Java Method References

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Understand how method (& constructor) references provide another foundational functional programming feature in Modern Java

Several examples showcase these Java 8+ function programming features

Learning Objectives in this Lesson

- Understand how method (& constructor) references provide another foundational functional programming feature in Modern Java
  - Also recognize the benefits of method references

Overview of Method References

Overview of Method References

- A compact, easy-to-read "handle" for a method that already has a *name*

In contrast, lambda expressions are *unnamed* blocks of code

Overview of Method References

- A compact, easy-to-read "handle" for a method that already has a *name*
  - It's shorthand syntax for a lambda expression that executes one method

See docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html

Overview of Method References

- There are four kinds of Java method references

Kind	Syntax	Method Reference	Corresponding Lambda Expression
1. Reference to a static method	ContainingClass::staticMethodName	String::valueOf	s -> String.valueOf(s)
2. Reference to an instance method of a particular object	containingObject::instanceMethodName	s::toString	() -> s.toString()
3. Reference to instance method of an arbitrary object of a given type	ContainingType::methodName	String::toString	s -> s.toString()
4. Reference to a constructor	ClassName::new	String::new	() -> new String()

See www.baeldung.com/java-method-references

Overview of Method References

- There are four kinds of Java method references

Kind	Syntax	Method Reference	Corresponding Lambda Expression
1. Reference to a static method	ContainingClass:: staticMethodName	String:: valueOf	s -> String .valueOf(s)
2. Reference to an instance method of a particular object	containingObject:: instanceMethodName	s::toString	() -> s.toString()
3. Reference to instance method of an arbitrary object of a given type	ContainingType:: methodName	String:: toString	s -> s.toString()
4. Reference to a constructor	ClassName::new	String::new	() -> new String()

References to Java static methods look like C++ pointer-to-static-method syntax

Overview of Method References

- There are four kinds of Java method references

Kind	Syntax	Method Reference	Corresponding Lambda Expression
1. Reference to a static method	ContainingClass:: staticMethodName	String:: valueOf	s -> String .valueOf(s)
2. Reference to an instance method of a particular object	containingObject:: instanceMethodName	s::toString	() -> s.toString()
3. Reference to instance method of an arbitrary object of a given type	ContainingType:: methodName	String:: toString	s -> s.toString()
4. Reference to a constructor	ClassName::new	String::new	() -> new String()

Overview of Method References

- There are four kinds of Java method references

Kind	Syntax	Method Reference	Corresponding Lambda Expression
1. Reference to a static method	ContainingClass:: staticMethodName	String:: valueOf	s -> String .valueOf(s)
2. Reference to an instance method of a particular object	containingObject:: instanceMethodName	s::toString	() -> s.toString()
3. Reference to instance method of an arbitrary object of a given type	ContainingType:: methodName	String:: toString	s -> s.toString()
4. Reference to a constructor	ClassName::new	String::new	() -> new String()

Overview of Method References

- There are four kinds of Java method references

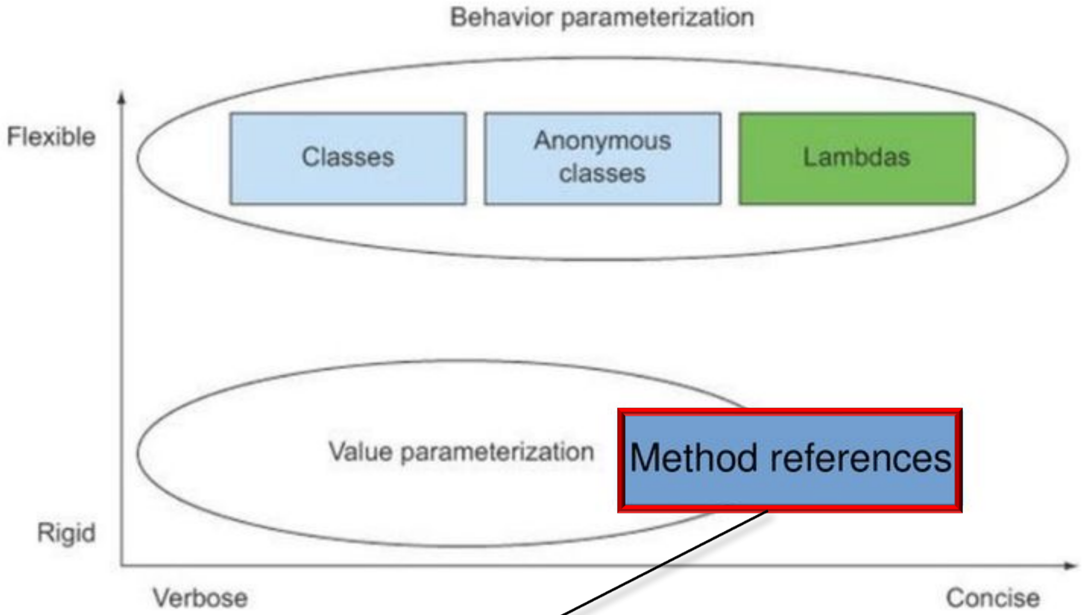
Kind	Syntax	Method Reference	Corresponding Lambda Expression
1. Reference to a static method	ContainingClass:: staticMethodName	String:: valueOf	s -> String .valueOf(s)
2. Reference to an instance method of a particular object	containingObject:: instanceMethodName	s::toString	() -> s.toString()
3. Reference to instance method of an arbitrary object of a given type	ContainingType:: methodName	String:: toString	s -> s.toString()
4. Reference to a constructor	ClassName::new	String::new	() -> new String()

See www.nextptr.com/tutorial/ta1314009273/uses-of-constructor-method-reference

Benefits of Method References

Benefits of Method References

- Method references are more concise than other behavior parameterizations



Java method references support more concise "behavior parameterization"

See blog.indrek.io/articles/java-8-behavior-parameterization

Benefits of Method References

- Method references are more concise than other behavior parameterizations, e.g.,

```
String[] nameArray = {"Barbara", "James", ..., "mary"};

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

vs

Lambda expressions are more concise than inner classes

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

See earlier lesson on "*Benefits of Java Lambda Expressions*"

Benefits of Method References

- Method references are more concise than other behavior parameterizations, e.g.,

```
String[] nameArray = {"Barbara", "James", ..., "mary"};

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

Benefits of Method References

- Method references are more concise than other behavior parameterizations, e.g.,

```
String[] nameArray = {"Barbara", "James", ..., "mary"};

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

Java IDEs can automatically create method references!

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

See stackoverflow.com/a/60858302

Benefits of Method References

- Method references are more concise than other behavior parameterizations, e.g.,

```
String[] nameArray = {"Barbara", "James", ..., "mary"};

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!

Benefits of Method References

- Method references are more concise than other behavior parameterizations, e.g.,

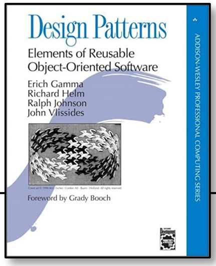
```
String[] nameArray = {"Barbara", "James", ..., "mary"};

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 comparisons is easy, a la the Strategy pattern


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

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

See en.wikipedia.org/wiki/Strategy_pattern

Benefits of Method References

- Method references are more concise than other behavior parameterizations, e.g.,

```
String[] nameArray = {"Barbara", "James", ..., "mary"};

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

It's good practice to use method references when you can!

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

See rules.sonarsource.com/java/RSPEC-1612

End of Overview of Java Method References