

Java 8 Lambda Expression Method Reference

“Code with Passion!”



Topics

- What is and Why Method Reference?
- 4 Types of Method References

What is and Why Method Reference?

What is a Method Reference?

- Method reference is just compact and more readable form of a lambda expression **for already existing methods**
 - > “::” operator is used for method reference
- Method signature (arguments and their types) of the existing method must match method signature of the functional interface

```
// Lambda expression  
MyFunctionallInterface myObject2 =  
    (name, age) -> MyClass.existingStaticMethod(name, age);  
myObject2.mySingleAbstractMethod("Jon", 77);
```

The method in the functional interface provides method signature

```
// Method reference  
MyFunctionallInterface myObject3 = MyClass::existingStaticMethod;  
myObject3.mySingleAbstractMethod("Jon", 77);
```

4 Types of Method References

4 Types of Method References

- #1: Reference to a static method - `ClassName::staticMethod`

`MyClass::myStaticMethod`

- #2: Reference to a constructor - `ClassName::new`

`String::new`

- #3: Reference to an instance method of a particular object - `objectOfClass::instanceMethod`

`myObject::myInstanceMethod`

- #4: Reference to an instance method of an arbitrary object of a particular type - `ContainingType::instanceMethod`

`String::compareIgnoreCase`

#1: Reference to a static method

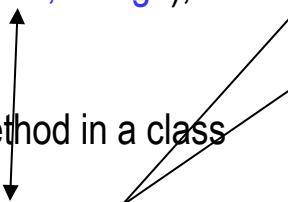
```
// Let's say you have defined a functional interface  
@FunctionalInterface  
interface MyFunctionalInterface {  
    public void mySingleAbstractMethod(String name, int age);  
}
```

```
// Let's also assume there is an existing static method in a class  
public class MyClass {  
    public static void existingStaticMethod(String name, int age) {  
        System.out.println(name + " is " + age + " years old.");  
    }  
}
```

```
// You would write a lambda expression as following if you want to use the existing static method of the class  
MyFunctionalInterface myObject2 = (name, age) -> MyClass(existingStaticMethod(name, age));  
myObject2.mySingleAbstractMethod("Jon", 77);
```

```
// The above lambda expression can be rewritten using a static method reference  
MyFunctionalInterface myObject3 = MyClass::existingStaticMethod;  
myObject3.mySingleAbstractMethod("Jon", 66);
```

The method signature (arguments & their types) of the existing method must match the method signature of the method of the functional interface



Lab:

**Exercise 1: Reference to
Static Method**

1612_javase8_lambda_method_reference.zip



#2 Reference to a Constructor - MyClass::new

- Similar to static method reference - only difference is that the method name is "new" (and, of course, the "new" constructor method always exists for a class)

```
// Let's say you have defined a functional interface  
@FunctionalInterface  
public interface MyFunctionalInterface {  
    public String createString(char[] cArray);  
}
```

String class has a constructor
method of
String(char[] cArray)

```
// You would write a lambda expression as following  
char[] charArray = { 'j', 'p', 'a', 's', 's', 'i', 'o', 'n' };  
MyFunctionalInterface fi1 = cArray -> new String(cArray);  
System.out.println(fi1.createString(charArray));
```

```
// The above lambda expression can be rewritten using a constructor method reference  
MyFunctionalInterface fi2 = String::new;  
System.out.println(fi2.createString(charArray));
```

Lab:

**Exercise 2: Reference to
Constructor**

1612_javase8_lambda_method_reference.zip



#3 Reference to instance method of a particular object

```
// Let's say you have defined a functional interface  
@FunctionalInterface  
interface MyFunctionalInterface {  
    public void mySingleAbstractMethod(String name, int age);  
}
```

```
// Let's also assume there is an existing instance method in a class  
public class MyClass {  
    public void existingInstanceMethod(String name, int age) {  
        System.out.println(name + " is " + age + " years old.");  
    }  
}
```

```
// Create an object that will be used within the body of lambda expression  
MyClass myObject = new MyClass();
```

```
// You would write a lambda expression as following  
MyFunctionalInterface myObject2 = (name, age) -> myObject.existingInstanceMethod(name, age);  
myObject2.mySingleAbstractMethod("Jon", 77);
```

```
// The above lambda expression can be rewritten using an instance method reference  
MyFunctionalInterface myObject3 = myObject::existingInstanceMethod;  
myObject3.mySingleAbstractMethod("Jon", 66);
```

Lab:

**Exercise 3: Reference to
Instance method**

1612_javase8_lambda_method_reference.zip



#4 Reference to instance method of an arbitrary object

- Produces a lambda with the **first argument being the object on which the method is called**
 - > This is the difference from #3 the instance method reference of a particular object
- The rest of the arguments are the arguments to the method
 - > This is the same with #3 the instance method reference of a particular object

```
// You would write a lambda expression as following  
Arrays.sort(stringArray, (String a, String b) -> a.compareToIgnoreCase(b));
```

```
// The above lambda expression can be rewritten using an instance  
// method reference of an arbitrary object – you use containing type, String in this case  
Arrays.sort(stringArray, String::compareToIgnoreCase);
```

#4 Reference to instance method of an arbitrary object

```
// Let's say you have defined a functional interface  
@FunctionalInterface  
interface MyInterface<T1, T2, R> {  
    R apply(T1 receiver, T2 argument1);  
}
```

```
// Let's also assume there is an existing instance method in a class  
public class MyClass {  
    Integer existingInstanceMethod(String s1) {  
        return s1.length();  
    }  
}
```

```
// Lambda expression  
MyInterface<MyClass, String, Integer> myfunction1 =  
    (MyClass myArbitrayObject, String s1) -> myArbitrayObject.existingInstanceMethod(s1);  
Integer result1 = myfunction1.apply(new MyClass(), "jpassion");
```

```
// Method reference  
MyInterface<MyClass, String, Integer> myfunction2 =  
    MyClass::existingInstanceMethod;  
Integer result2 = myfunction2.apply(new MyClass(), "jpassion");
```

Lab:

**Exercise 4: Reference to Instance
Method of an Arbitrary Object**

1612_javase8_lambda_method_reference.zip



Code with Passion!

