

Lambdas

Method References

// shorter way of writing lambda expression

@FunctionalInterface

interface Animal {

public void speak(String s);

}

public class MyClass {

public static void shout(String s, Animal a) {

a.speak(s);

}

public static void main(String[] args) {

Animal myAnimal = s -> System.out.println(s);

shout("woof!", myAnimal);

}

}

System.out::println



Method Reference and context

- Java already knows number of parameters of abstract method
 - so they are automatically inserted without the need of explicitly list them
- for example:

```
s -> System.out.println(s)
```

- this lambda is implementing method `speak(String s)`
 - Java knows there is only one parameter in question
 - so you can just as well omit it and write it shorter:

```
System.out::println
```

```
// using method reference with static method, e.g. Math.min()
@FunctionalInterface
interface Calculator {
    public int minimum(int a, int b);
}

public class MyClass {
    public static void main(String[] args) {
        Calculator lambda = (a, b) -> Math.min(a, b);
        Calculator methodRef = Math::min;
        System.out.println(lambda.minimum(-3, 1));
        System.out.println(methodRef.minimum(-3, 1));
    }
}
```

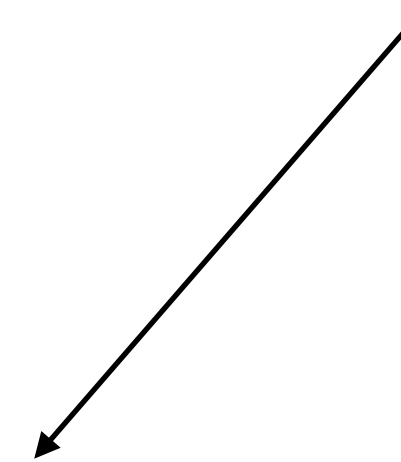
EQUIVALENT

```
@FunctionalInterface
interface Calculator { public double path(double t); }

class Gravity {
    public static double freeFall(double t) {
        final double g = 9.81;
        return 0.5 * g * t*t;
    }
}

public class MyClass {
    public static void main(String[] args) {
        Calculator methodRef = Gravity::freeFall;
        System.out.println(methodRef.path(10));
    }
}
```

t -> Gravity.freeFall(t)



// calling method reference on an Object

@FunctionalInterface

interface Checker {

public boolean check();

}

public class MyClass {

public static void main(String[] args) {

String s = "John Wayne";

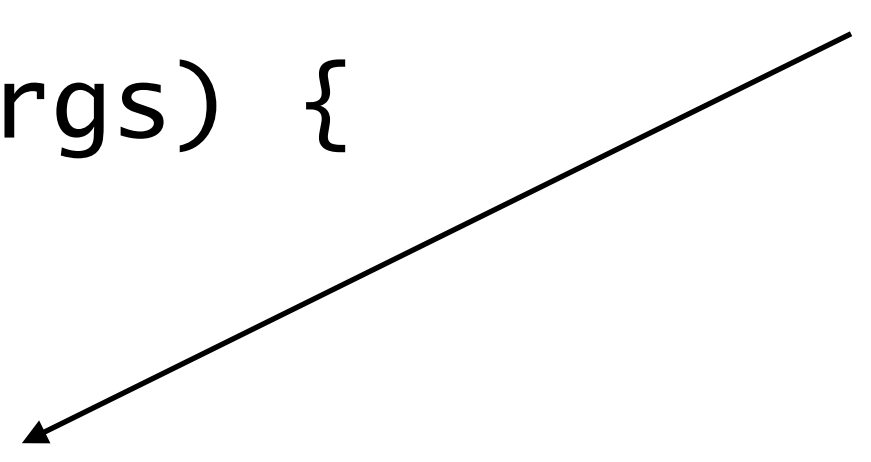
Checker methodRef = s::isEmpty;

System.out.println(methodRef.check());

}

}

() -> s.isEmpty();

A black arrow points from the boxed text 's::isEmpty;' in the code block to the boxed text '() -> s.isEmpty();'.

```
// calling method reference on a parameter
```

```
@FunctionalInterface
```

```
interface Checker {
```

```
    public boolean check(String s);
```

```
}
```

```
public class MyClass {
```

```
    public static void main(String[] args) {
```

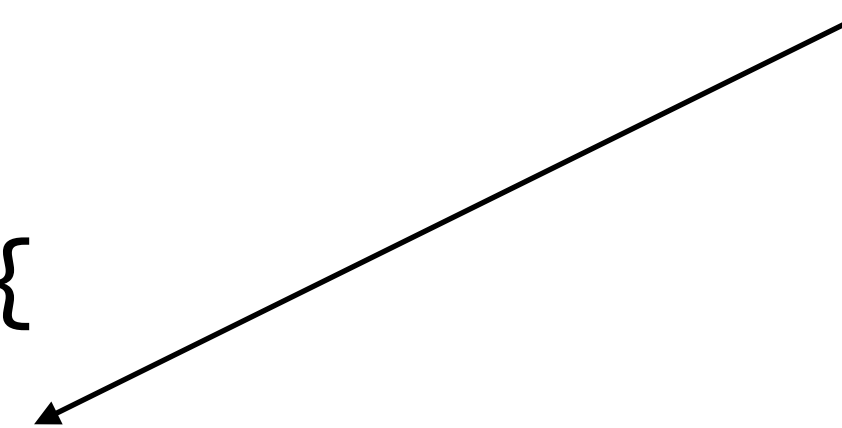
```
        Checker methodRef = String::isBlank;
```

```
        System.out.println(methodRef.check("  "));
```

```
    }
```

```
}
```

s -> s.isBlank();



```
// calling constructor reference
```

```
@FunctionalInterface
```

```
interface Teller {
```

```
    String tellName(String name);
```

```
}
```

```
public class MyClass {
```

```
    public static void main(String[] args) {
```

```
        Teller methodRef = String::new;
```

```
        System.out.println(methodRef.tellName("John Wayne"));
```

```
    }
```

```
}
```

s -> new String(s);



Summary

Lambda	Method Reference
<code>s -> System.out.println(s)</code>	<code>System.out::println</code>
<code>(a, b) -> Math.min(a, b)</code>	<code>Math::min</code>
<code>t -> Gravity.freeFall(t)</code>	<code>Gravity::freeFall</code>
<code>() -> s.isEmpty()</code>	<code>s::isEmpty</code>
<code>s -> s.isBlank()</code>	<code>String::isEmpty</code>