Invoking a Lambda Expression on Objects and Primitive Types

José Paumard
PHD, JAVA CHAMPION, JAVA ROCK STAR
@JosePaumard https://github.com/JosePaumard

Agenda

Let us talk about performances!

How Fast Lambda Expressions Are?

Lambdas are not instances of anonymous classes!

The compiled code is different

And uses invokedynamic

Performances are 60x better

But you can make them even faster

```
Comparator<Integer> cmp =
        (i1, i2) -> Integer.compare(i1, i2);

int compared = cmp.compare(10, 20);
```

A classical comparator to compare ints
in the classical way

What is happening under the hood when this comparator is called?

```
public interface Comparator<T> {

    int compare(T t1, T t2);
}

public interface Comparator<Integer> {

    int compare(Integer t1, Integer t2);
}
```

Boxing!

= and int is “boxed” in an Integer, transparently

... and this has a cost

To overcome that, a set of functional interfaces has been added:

- IntPredicate
- LongSupplier
- IntFunction<T>
- LongToIntFunction

Demo

Let us see some code!

And work with those specialized lambdas

Module Wrap Up

What did you learn?

Lambda expressions are fast!

How to use specialized lambdas when working with primitive types