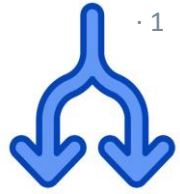


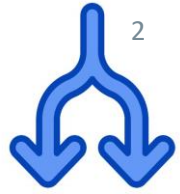
# Practical Concurrent and Parallel Programming

## Introduction to lambdas in Java

Jørgen Staunstrup

# Motivation





In week 5 of PCPP you will see code like this:

```
Mark6( . . . , i -> multiply(i));
```

**`Mark6`** is a Java method with several parameters, the last one

```
i -> multiply(i)
```

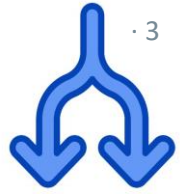
is a lambda expression defining a function

This presentation gives an introduction to lambdas in Java

https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

# Passing parameters



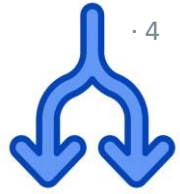


```
class parameterExample {
  public static void main(String[] args) { new parameterExample(); }
  public parameterExample() {
    objectEx o= new objectEx();

    int i= 5;
    System.out.println("Before update:"+i);
    updateint(i);
    System.out.println("After update:"+i);

    o.setli(5);
    System.out.println("Before update:"+o.get());
    updateObject(o);
    System.out.println("After update:"+o.get());
  }
  public void updateObject(objectEx p) { p.setli(7); }
  public void updateint(int p) { p= 7; }
```

```
class objectEx {
   private int li;
   public void setli(int v) { li= v;}
   public int get() {return li;}
}
```

# Different kinds of parameters





```
String w= "      ...          ";

if Character.isLowerCase(w.charAt(0)) {   ...
```

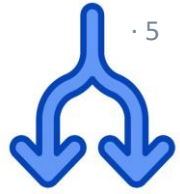
What is `w.charAt(0)`?

What is `w`?

```
MakeLowerCase(w);
```

# Different kinds of parameters





```
String w= "      ...          ";

if Character.isLowerCase(w.charAt(0)) {   ...
```

w.charAt(0) is a value

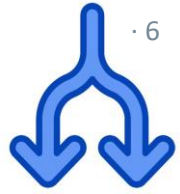
w is an object

```
MakeLowerCase(w);
```

What is charAt?

# Different kinds of parameters





```
String w= "      ...          ";

if Character.isLowerCase(w.charAt(0)) {   ...
```

w.charAt(0) is a value

w is an object

```
MakeLowerCase(w);
```

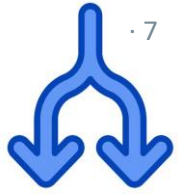
charAt is a function

How can we transfer the function "**charAt" itself?**

**by using a lambda (expression)**

# Lambda examples





In week 2

```
new Thread( () -> { vrw.reader(); } )
```

In week 5

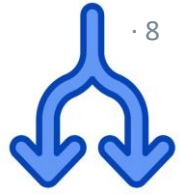
```
Mark6( . . . , i -> multiply(i)  );
```

In week 7

```
count= readWords(filename)  // makes a stream of words
  .filter( w -> w.length()>1 )
  .count(); // H
```

# Lambda syntax (Java 8)

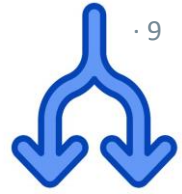




```
import java.util.function.Function;

class LambdaExample {
  public static void main(String[] args) { new LambdaExample(); }

  public LambdaExample() {
    System.out.println("I: "+increment(f));
  }

  Function<Integer, Integer> f = (x) -> x+1;  // f(x) = x+1

  private static int increment(Function<Integer, Integer> add1) {
    return add1.apply(2);  //   f(2)
  }
}
```

# Lambda expressions





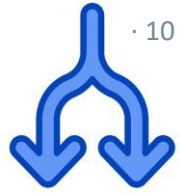
Argument type Result type Lambda expr.

- One argument
  - `Function<Integer, Integer> f = (x) -> x+1`
  - $f : \mathbb{Z} \rightarrow \mathbb{Z} \mid f(x) = x + 1$
  - $f(1)$ <-> `f.apply(1)`
- Two arguments
  - `BiFunction<Integer,Integer,Integer> f = (x,y) -> x+y`
  - $f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z} \mid f(x, y) = x + y$
  - $f(1,2)$ <-> `f.apply(1,2)`

https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html

# Lambda expressions (2)





- Zero arguments
  - `Supplier<Integer> f = () -> 2`
  - $f : \mathbb{Z} |\ \ f() = 2$
  - $f()$ <-> `f.get()`

- No return type (`void`)
  - `Consumer<Integer> f = (x) -> System.out.println(x);`
  - $f(2)$ <-> `f.accept(2)`

  - `() -> { vrw.reader(); }`

Examples in Week07: code-exercises/.../FuncionExamples.java

# Method references

The methods of an object may also be referenced as follows

- `Class::method`
  - `BiFunction<String,Integer,Character> f = String::chartAt`
  - `f.apply(s,i) <-> s.charAt(i)`
  - `Function<Person,String> f = Person::getName`
  - `System.out::println`
  - …
- <Object instance>::method
  - `Function<Integer,Character> f = "01234"::charAt`
  - `f.apply(i) <-> "01234".charAt(i)`

javaprecisely-3rd-draft-streams.pdf (section 11.14) and

https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html

# Try it yourself



The exercises for week 7 (exercises07.pdf) has an extra (non-mandatory exercise:

*Not mandatory*

If you are already comfortable with Java lambdas, you may skip this exercises. If you are not, please try to solve it and turn in your solution.

**Exercise 7.1** You may use this Java skeleton as a starting point of the exercise.

```
import java.util.function.Function;
class LambdaExample {
  public static void main(String[] args) { new LambdaExample(); }
```

If lambdas is a new concept for you, try to solve this exercise

Use the learnIT forum if you have questions