**Java 8 Lambdas and Streams**

https://learning.oreilly.com/videos/java-8-lambdas/9781771374743/9781771374743-video235322?autoplay=false

Why Java 8

Advances in processing power

Parallel processors

Big data applications

* Passing behavior
  + Anonymous classes
  + Functional programming support
  + Cannot pass a methos to a method
* Enhancing the libraries
  + Libraries need makeover
  + Interface unlocking
* Outcome
  + Lambdas – for representing the behavior
  + Streams
  + Interface evolution

Brief introduction to java 8 new features

* Lambdas
  + Functional style
  + Behavior parameterization
* Interface unlocking
  + Concrete methods
* Functional interfaces
  + Interface with only one abstract methos
  + Lambda expression
  + These interfaces will be target type for lambda expression
* Default and static methods in an interface
* New Date and time
  + JODA time influence

What are Lambdas?

* Everything is an object
  + Class and objects
  + Methods are tied to the class
  + Anonymous classed are functions
  + Mutability is inherent
* Lambdas are
  + Anonymous functions
  + () -> sendEmail();
  + (greeting) -> System.out.prontln(“Hello”+s);
  + Represents behavior
  + Compact code blocks that can be Passed to other methods
* Lambda Systax
  + Simple method signature - Input arguments, body and return
  + (input\_args) -> body
  + Left hand side is input-------- right hand side is logic
  + Example
    - (integer i) -> i\*I;
    - (string a) -> notify(a);
* Simple Lambdas
  + Concatenating two strings
    - (String s1, String s2) -> s1+s2;
  + Square of a number
    - (Integer num) -> num \* num;
  + Complex business logic
    - (List<Trades> trades) -> {

persist (trades);

notify();

}

Creating a Lambda

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

SpringApplication.*run*(SpringstartApplication.**class**, args);

**new** SpringstartApplication().testGreeting("Harry", (String s) -> "Hello"+s);

new SpringstartApplication().testGreeting("Pingu",

(String p) -> !p.isEmpty() ? "Howdy " + p : "Did you miss something");

}

**interface** Greeting{ //Single abstract method in the interface, hence this can be used for lambda expression

**public** String sayHello(String g);

}

**public** **void** testGreeting(String a, Greeting g) {

String result = g.sayHello(a);

System.***out***.println("Result "+result);

}

Lambda Syntax

(Input\_Parameter) -> {body}

* Return Values
  + Implicit return
  + Explicit return
  + (i) -> i\*I //no type declaration
  + (String s1, String s2) -> s1 + s2 //implicit return
  + (Trades<Trade> trades , Price p) -> { //Logic}
  + (String name) -> System.out.println(“Hello”+name)
  + (name) -> System.out.println(“Hello”+name)
  + name -> System.out.println(“Hello”+name)
  + () -> autopilot() //no parameters, no return type
  + (Trade t, Price p) -> {
    - p.applyPricing(t)
    - return t;
  + }

Target typing

* I -> I\*I
* Format name as email

//(String name) -> "name"+"abc@asd.com"

**public** **interface** Email{

String constructEmail(String name);

}

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

SpringApplication.*run*(SpringstartApplication.**class**, args);

Email email = (String name) -> name+"asd.com";

}

=======================================================

//(String name) -> "name"+"abc@asd.com"

**public** **interface** Email{

String constructEmail(String name);

}

**public** String getEmail(String name, Email email) {

String email1 = email.constructEmail(name);

System.***out***.println(email1);

**return** email1;

}

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

SpringApplication.*run*(SpringstartApplication.**class**, args);

Email email = (String name) -> name+"asd.com";

**new** SpringstartApplication().getEmail("Anic", (String name) -> name+"asd.com");

}

========================================

Capturing Lambda

@FunctionalInterface

**interface** Multiplier{

**int** multiply(**int** a, **int** b);

}

**public** **int** multiplierService(**int** a, **int** b, Multiplier multiplier) {

**int** abc = multiplier.multiply(a, b);

**return** abc;

}

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

SpringstartApplication springstartApplication = **new** SpringstartApplication();

**int** mul = springstartApplication.multiplierService(10, 5, (a,b) -> a\*b);

System.***out***.println(mul);

**int** div = springstartApplication.multiplierService(10, 5, (a,b) -> a/b);

System.***out***.println(div);

**int** add = springstartApplication.multiplierService(10, 5, (a,b) -> a+b);

System.***out***.println(add);

}