

Forum:[Hack Forums](#)

Thread: [Lambdas in Java \[LEGENDS\]](#)

SIM-2015410 - September 4, 2015, 5:54 pm

[img]http://s9.postimg.org/57pm7cwyr/LEGENDS_2_0_HEADER.png[/img]

Lambdas in Java 8

Introduction:

Java is a little late to the party, but in Java 8 it is now possible to use Lambdas. These allow us to write methods and complex collection processing algorithms using simpler, cleaner and easier to read code. Lambdas can also improve performance on multicore CPUs.

[img]http://s9.postimg.org/jzubyp2v3/LEGENDS_2_0_DELIMITER.png[/img]

OOP vs Functional

Java is pretty much an OOP language (there are some little details which say it isn't pure OOP, but it is close). Lambdas introduce a more functional element to the language. They are an abstraction over behaviour rather than data.

The lambda code is easier to read because it expresses intent of its logic not mechanics of how it's achieved, and lambdas allow us to pass functions around easily allowing for more ability to write lazy code (evaluated only when needed).

[img]http://s9.postimg.org/jzubyp2v3/LEGENDS_2_0_DELIMITER.png[/img]

Before Java 8

Before Java 8, it was common to use an anonymous inner class to associate behaviour with an event like a mouse-click. For example:

```
Code button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent event){
        System.out.println("Hello World!");
    }
});
```

In this case, we are giving the button an object to perform the action.

[img]http://s9.postimg.org/jzubyp2v3/LEGENDS_2_0_DELIMITER.png[/img]

Lambdas

Lambdas allow us to associate behaviour with a button click like this:

```
Code button.addActionListener(event -> System.out.println("Hello World"));
```

The contents of the bracket is the Lambda expression, it is like a function without a name. Let's break it down:

event is the name of the parameter that this lambda accepts. Notice that we do not have to declare the type of the parameter, the compiler can infer this. Types are purely optional.

We can also accept multiple parameters, by surrounding them with brackets. Brackets are optional for single parameters.

The arrow -> separates the parameters from the body.

The stuff to the right of the arrow, System.out.println("Hello World");, is the function body. This can

optionally have a return statement, the compiler will automatically return the value if the body contains a single statement. We can surround the body with curly braces if you want, but you don't have to if it only contains one statement (same as if statements). You have to have curly braces to use a return statement.

Here is an example with all the optional syntax added for demonstration:

```
Code (int a, int b) -> { return a * b; };
```

It is also possible to write a function which takes no parameters, by leaving the brackets empty:

```
Code () -> System.out.println("Hello World!");
```

Because of the functional nature of lambdas, they also come with a rule: you cannot change the value of a variable that is used in a lambda expression, it must be treated as if it were final. If you try and assign a new value to a variable that is used in a lambda expression, it will fail to compile, for example:

```
Code String name = getUsername();
```

```
name = formatUsername(name);
```

```
button.addActionListener(event -> System.out.println(name));
```

[img]http://s9.postimg.org/jzubyp2v3/LEGENDS_2_0_DELIMETER.png[/img]

Storing Lambdas

You can store lambdas in variables, for example Runnable, ActionListener, BinaryOperator, etc.

Example:

```
Code Runnable coolLambda = () -> System.out.println("Hello World!");
```

coolLambda.run(); Usually the number of arguments the lambda accepts will depend on the type you store it in. For example, ActionListener lambdas take one parameter (an action).

[img]http://s9.postimg.org/jzubyp2v3/LEGENDS_2_0_DELIMETER.png[/img]

Functional Interfaces

A functional interface is an interface with a single abstract method that is used as the type for a lambda expression. For example:

```
Code public interface ActionListener extends EventListener {
```

```
    public void actionPerformed(ActionEvent event);
```

```
} In this example, the abstract method actionPerformed is used to represent an action that takes one parameter and returns no result.
```

Other examples of functional interfaces are:

Predicate<T> - takes an argument of type T and returns true or false.

Function<T,R> - takes an argument of type T and returns type R.

BinaryOperator<T> - takes 2 arguments of type T and returns a value of type T

Example of Predicate<T> :

Code Predicate<Integer> moreThanTen = x -> x > 10; the type of the parameter in the lambda is inferred as an Integer, the return type must be a boolean

[img]http://s9.postimg.org/jzubyp2v3/LEGENDS_2_0_DELIMETER.png[/img]

This tutorial was written by MeshCollider ?

Angel Beats?? - September 4, 2015, 9:26 pm

No one replied to your thread cos Java is shit.

Delirium™ - September 4, 2015, 9:26 pm

MeshCollider has been posting some very high quality tutorials lately. Thanks a lot for this contribution.

[img]http://orig03.deviantart.net/a7ce/f/2010/239/d/f/forever_alone_by_foreveraloneplz.png[/img]

SIM-2015410 - September 4, 2015, 9:27 pm

(Sep 4, 2015 09:26 PM)Angel Beats?? Wrote: [\[CITE\]](#) No one replied to your thread cos Java is shit. Thanks for your feedback ?

Balding - September 4, 2015, 9:28 pm

(Sep 4, 2015 09:26 PM)Angel Beats?? Wrote: [\[CITE\]](#) No one replied to your thread cos Java is shit. #JavaMasterRace

Shaddup anal beads <3

TheLordCrazy - September 5, 2015, 3:11 am

Thanks you! That's how you explain something with simple words. Not like all those books called "blablaba for Beginners", "blablala for complete newbs" etc. I don't want you to fuckin' use 138948138913489 other words before you explain what's going on, I just want you to say it in a understandable way. So once again thanks for this tutorial!

SIM-2015410 - September 5, 2015, 3:56 am

(Sep 5, 2015 03:11 AM)TheLordCrazy Wrote: [\[CITE\]](#) Thanks you! That's how you explain something with simple words. Not like all those books called "blablaba for Beginners", "blablala for complete newbs" etc. I don't want you to fuckin' use 138948138913489 other words before you explain what's going on, I just want you to say it in a understandable way. So once again thanks for this tutorial! No problem, I am glad you like it ?

Reval - September 5, 2015, 4:54 am

Nice tutorial. Thanks man!

+Haze - September 5, 2015, 5:01 am

(Sep 4, 2015 09:26 PM)Angel Beats?? Wrote: [\[CITE\]](#) No one replied to your thread cos Java is shit. I laugh at all the programming posts you make, because frankly you post shit all the time.