

# Lambda Expressions

# Lambda Expressions

Feature added in JDK 8.

**Lambda expression:** Anonymous method.

Cannot execute on its own, instead is used to implement a method defined by functional interface.

**Functional interface:** Interface which contains only one abstract method.

**Abstract method:** Body with no implementation.

Functional interface represents a single action.



# Lambda Expression.

Lambda operator: ->

Example: (left side) -> (right side)

Left side: parameters required by the lambda expression.

Right side: lambda body, specifies actions of the lambda expression.

Lambda bodies can be of two type: single expression or block of code.



# Lambda Expression Example

```
int result(){ return 100 }
```

The above method can be converted into lambda expression as:

```
() -> 100
```

Empty parameter list, lambda operator & code.



# Lambda Expression Example

->  $100 * 200$  : Returns multiplication of two integers.

(n) ->  $n * 100$ : Accepts parameter n, multiples it with 100.

(a,b)->  $a*b$  : Passing multiple parameters.



# Functional interface


An interface which contains only one abstract method.

But aren't all methods of an interface abstract?

They were before Java 8.

Since Java 8, default implementation for method declaration in an interface is possible.

Hence currently, an interface method is abstract only if it does not specify an implementation.



# Functional interface example:

```
Interface Number(){  
    Int displayNumber();  
}
```

Does Message interface have only one method: YES

Is the method abstract : YES

This implies that Message is a functional interface.



# How to execute a lambda expression?

A lambda expression cannot be executed on its own.

Instead, it is an implementation of the abstract method in the functional interface.

Interface: `Interface Number(){`

`Int displayNumber();`

`}`

Lambda expression: `() -> 100`





# How to execute a lambda expression?

```
package com.company;  
  
interface Number{  
  
    int displayNumber();  
}
```

```
public static void main(String[] args){  
  
    Number n;  
    n = ()-> 100;  
    System.out.println(n.displayNumber());  
  
}
```

1. Create an interface.
2. Create a reference to interface instance.
3. Assign lambda expression to that interface instance.
4. Thus, lambda expression transforms the code into an object "n".

Let's write some code.....

