# Java Lambda Expressions

‹ Previous
Next ›

## Java Lambda Expressions

Lambda Expressions were added in Java 8.

A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

## Syntax

The simplest lambda expression contains a single parameter and an expression:

```
parameter -> expression
```

To use more than one parameter, wrap them in parentheses:

```
(parameter1, parameter2) -> expression
```

Expressions are limited. They have to immediately return a value, and they cannot contain variables, assignments or statements such as `if` or `for`. In order to do more complex operations, a code block can be used with curly braces. If the lambda expression needs to return a value, then the code block should have a `return` statement.

```
(parameter1, parameter2) -> { code block }
```



# Using Lambda Expressions

Lambda expressions are usually passed as parameters to a function:

## Example

Use a lamba expression in the `ArrayList`'s `forEach()` method to print every item in the list:

```
import java.util.ArrayList;
```

```
public class Main {
  public static void main(String[] args) {
    ArrayList<Integer> numbers = new ArrayList<Integer>();
    numbers.add(5);
    numbers.add(9);
    numbers.add(8);
    numbers.add(1);
    numbers.forEach( (n) -> { System.out.println(n); } );
  }
}
```

Try it Yourself »

Lambda expressions can be stored in variables if the variable's type is an interface which has only one method. The lambda expression should have the same number of parameters and the same return type as that method. Java has many of these kinds of interfaces built in, such as the `Consumer` interface (found in the `java.util` package) used by lists.

## Example

Use Java's `Consumer` interface to store a lambda expression in a variable:

```
import java.util.ArrayList;
import java.util.function.Consumer;

public class Main {
  public static void main(String[] args) {
    ArrayList<Integer> numbers = new ArrayList<Integer>();
    numbers.add(5);
    numbers.add(9);
    numbers.add(8);
    numbers.add(1);
    Consumer<Integer> method = (n) -> { System.out.println(n); };
    numbers.forEach( method );
  }
}
```

Try it Yourself »

To use a lambda expression in a method, the method should have a parameter with a single-method interface as its type. Calling the interface's method will run the lambda expression:

## Example

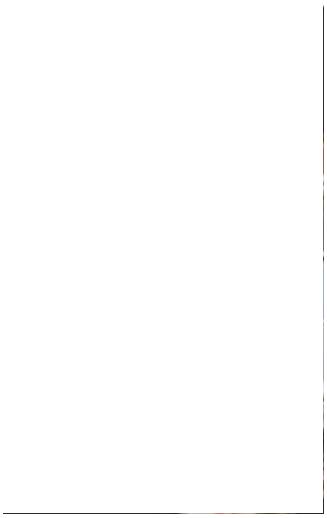
Create a method which takes a lambda expression as a parameter:

```
interface StringFunction {
  String run(String str);
}

public class Main {
  public static void main(String[] args) {
    StringFunction exclaim = (s) -> s + "!";
    StringFunction ask = (s) -> s + "?";
    printFormatted("Hello", exclaim);
    printFormatted("Hello", ask);
  }
  public static void printFormatted(String str, StringFunction format) {
    String result = format.run(str);
    System.out.println(result);
  }
}
```

Try it Yourself »

‹ Previous Next ›



## Top Tutorials

PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

## Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

## Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

## Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate