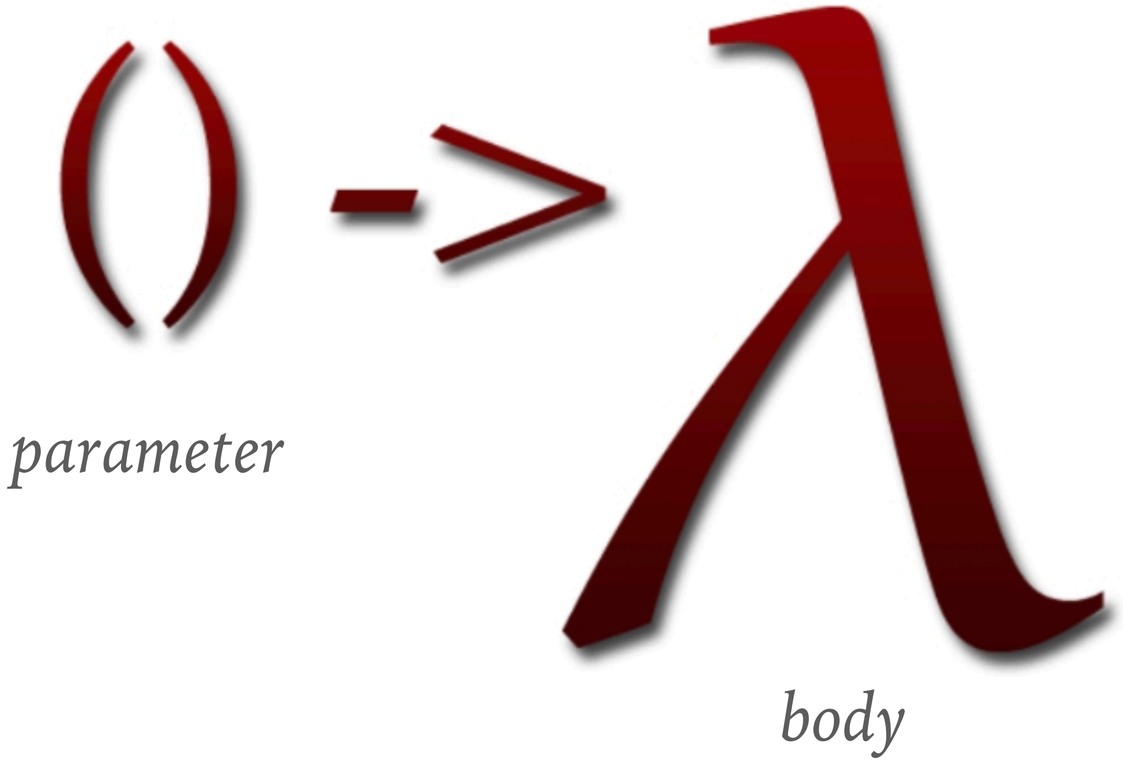
LAMBDA EXPRESSION

A **lambda expression** is a short block of code which takes in parameters and returns a value.



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
```

Why do we need Lambda expression?

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.

Lambda helps to write a method and use it immediately, so it saves time and effort. (Especially if we use this method only once)

Lambda Expression Java 8

Example:

```
ArrayList<Integer> nums3 = new ArrayList<>(Arrays.asList(1, 5, 2, 3, 5, 2, 1, 8));

//remove all the numbers that are less than 5
nums3.removeIf(n -> n < 5); // n will be every element, removeIf is a method
System.out.println(nums3);
```

RemoveNums

```
/Library/Java/JavaVirtualMachines/jdk-11.0.12.jdk/Contents/Home/bin/java -javaagent:/Users/a
[5, 5, 8]
```

In order to do more complex operations, a code block can be used with curly braces.

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

Example

Use a lambda 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); } );
  }
}
```

Lambda Syntax

- No arguments:
```
() -> System.out.println("Hello")
```
- One argument:
```
s -> System.out.println(s)
```
- Two arguments:
```
(x, y) -> x + y
```
- With explicit argument types:
```
(Integer x, Integer y) -> x + y
```
- Multiple statements:
```
(x, y) -> {
    System.out.println(x);
    System.out.println(y);
    return (x + y);
}
```