Lambda expression is the best feature came with Java 8.

Anonymous function is nameless or unknown function. A method which do not have any name or modifier

What is a Lambda expression? An expression through which we can represent an Anonymous function.

|  |  |  |
| --- | --- | --- |
| Parameter | Expression | body |
| () | -> | System.out.println("Some Statement"); |
| (I,j) | -> | I+j; |
|  |  |  |

In below class we represented m1 method as lambda expression, but how can we call it (see in main method we can call m1 method using Class instance. But how can we call lambda expression method as it doesn’t have any name.

public class Example001{

    public void m1() {

        System.out.println("Some Statement");

    }

    () -> System.out.println("Some Statement");

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

        new Example001().m1();

    }

}

We can use Functional interface to call to call lambda expression.

**Functional Interfaces in Java**

A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, [lambda expressions](https://www.geeksforgeeks.org/lambda-expressions-java-8/) can be used to represent the instance of a functional interface. A functional interface can have any number of default methods. Runnable, ActionListener, Comparable are some of the examples of functional interfaces.

Functional Interface is additionally recognized as Single Abstract Method Interfaces. In short, they are also known as SAM interfaces. Functional interfaces in Java are the new feature that provides users with the approach of fundamental programming.

@FunctionalInterface

public interface MyInterface{

    public void m1();

}

public class Example001{

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

       MyInterface i =() -> System.out.println("Some Statement");

       i.m1();

    }

}

Java 8 features

Differences between Java 8 Map() Vs flatMap() :

| **map()** | **flatMap()** |
| --- | --- |
| It processes stream of values. | It processes stream of stream of values. |
| It does only mapping. | It performs mapping as well as flattening. |
| It’s mapper function produces single value for each input value. | It’s mapper function produces multiple values for each input value. |
| It is a One-To-One mapping. | It is a One-To-Many mapping. |
| Data Transformation: From Stream to Stream | Data Transformation: From Stream<Stream to Stream |
| Use this method when the mapper function is producing a single value for each input value. | Use this method when the mapper function is producing multiple values for each input value. |