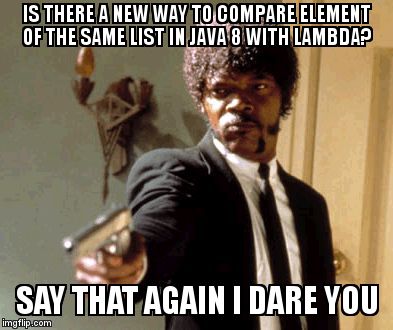
What Is Lambda Expression? and Why it is Important?

Can You Tell This?????????????????????



***Which topic we will cover in this article:***

1. What is lambda
2. Why lambda introduced in Java 8
3. What is functional interface in lambda expression.
4. How to use lambda
5. Example of lambda
6. Advantages of lambda

This feature is introduced in Java 8 with another features as well likw Stream API, Date API, etc.

***What is Lambda?***

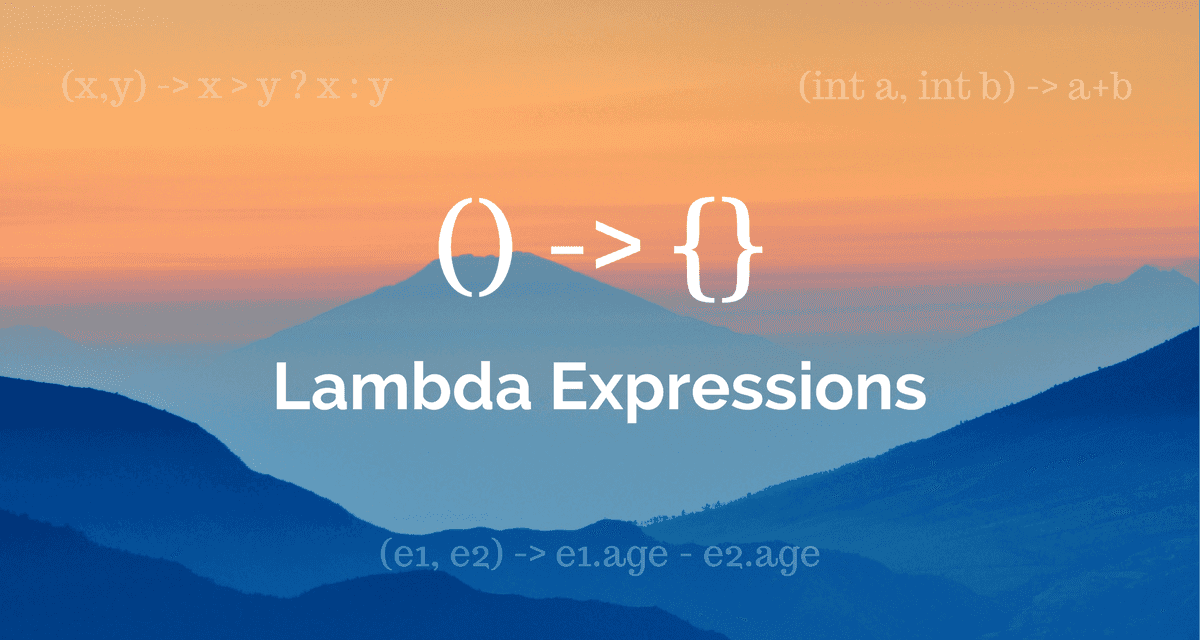
This expression is added into java 8 only, before the version of Java 8 don’t have this feature.

Lambda expression is short block of code which takes in parameter’s and also return’s the value.

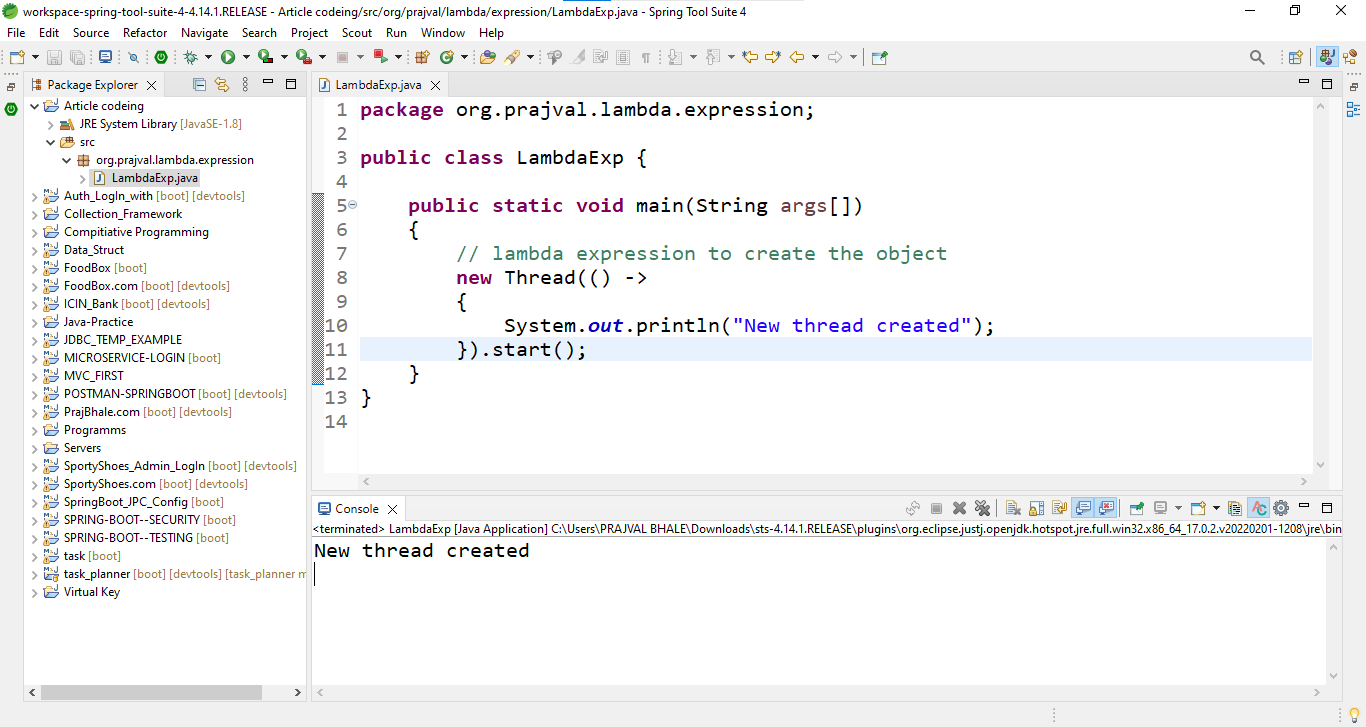
This is much more similar to the method’s, but lambda don’t need any name and they implemented into the method body right.

*Syntax:*

Parameter -> expression



Onwards the Java 8 we can assign Lambda expression to it’s functional interface object is like following:



**package** org.prajval.lambda.expression;

**public** **class** LambdaExp {

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

{

// lambda expression to create the object

**new** Thread(() ->

{

System.***out***.println("New thread created");

}).start();

}

}

OUTPUT:

New thread created

***Why lambda introduced in Java 8***

There are various reasons for addition of lambda expression in Java 8 but the most beneficial of them is that we can easily distribute processing of collection over multiple threads.

Prior to Java 8, if the processing of elements in a collection had to be done in parallel, the client code was supposed to perform the necessary steps and not the collection.

In Java 8, using ***lambda expression and Stream API*** we can pass processing logic of elements into methods provided by collections and now collection is responsible for parallel processing of elements and not the client.

Java 8 provide, support for lambda expressions only with functional interfaces.

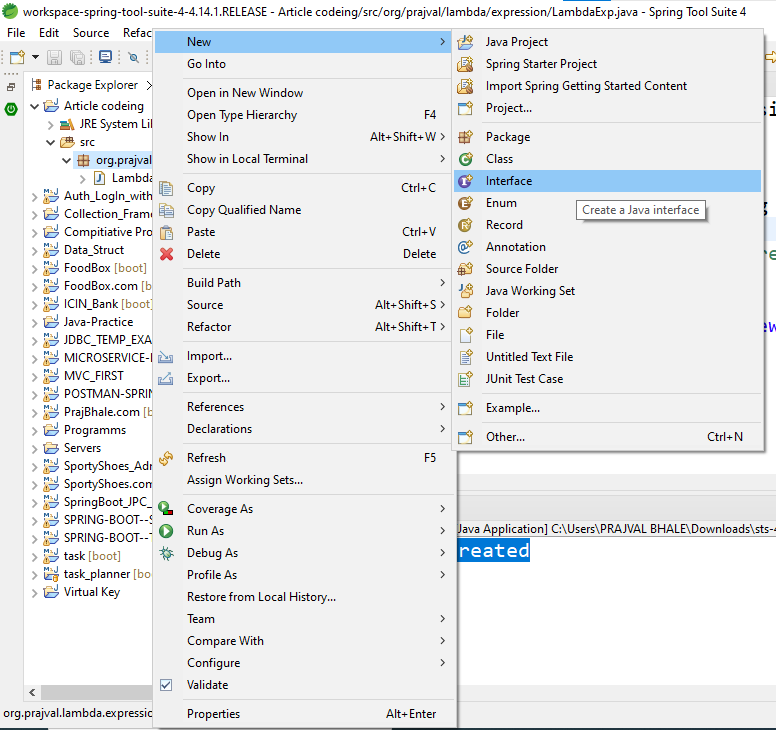
Any Interface with single abstract method is called ***Functional Interface***. there is only one abstract method, there is no confusion in applying the lambda expression to that method.

***What is functional interface in lambda expression?***

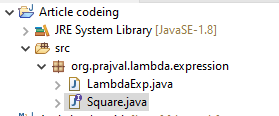
Which interface contains only one abstract method. They have only one functionality to exhibit. Some real world examples of functional interface is: ***Runnable, ActionListener, Comparable.***

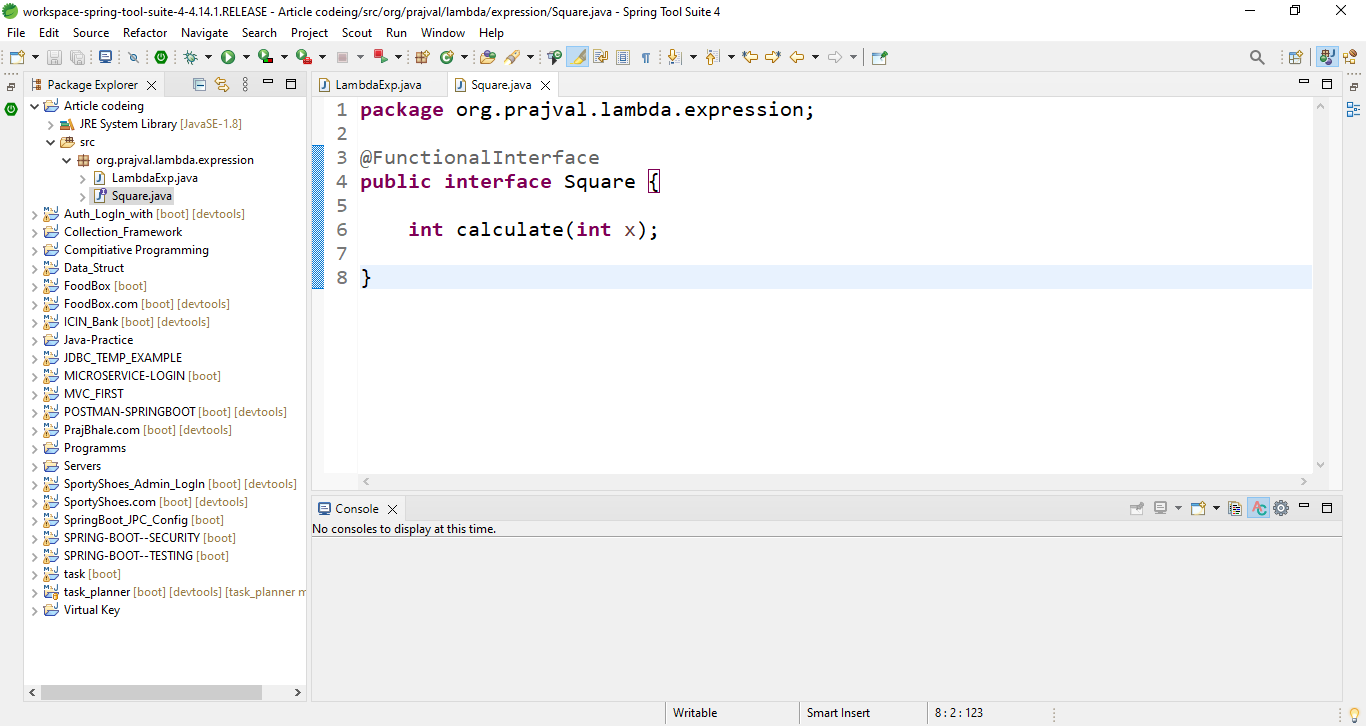
They are also know as ***SAM Method***, mean’s **Single Abstract Method Interface.**

How to make functional interface and how to implement into the lambda expression, see in example:



After interface creation successfully,





***Use and Example of Lambda Expression:***

Now we successfully Function Interface is create, now main question is how to implement this interface in our lambda expression?

**package** org.prajval.lambda.expression;

@FunctionalInterface

**public** **interface** Square {

**int** calculate(**int** x);

}

**package** org.prajval.lambda.expression;

**public** **class** TestInterface{

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

{

**int** a = 5;

//lambda exp. to define the calculate method

Square s = (**int** x) -> x \* x;

**int** ans = s.calculate(a);

System.***out***.println(ans);

}

}

***Advantage’s of Lambda Expression:***

1. Few Lines of Code: this is the main advantage of lambda expression.
2. Also already we see collection for processing the elements either in a sequential or parallel manner functions are passed to collection method.
3. By using lambda and expression and stream API, we can achieve higher efficiency in case of bulk operation’s.