Java Lambda Expressions

Lambda expression is a new and important feature of Java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.

Java lambda expression is treated as a function, so compiler does not create .class file.

Functional Interface

Lambda expression provides implementation of *functional interface*. An interface which has only one abstract method is called functional interface. Java provides an anotation @*FunctionalInterface*, which is used to declare an interface as functional interface.

Why use Lambda Expression

1. To provide the implementation of Functional interface.
2. Less coding.

Java Lambda Expression Syntax

(argument-list) -> {body} ;

Java lambda expression is consisted of three components.

**1) Argument-list:** It can be empty or non-empty as well.

**2) Arrow-token:** It is used to link arguments-list and body of expression.

**3) Body:** It contains expressions and statements for lambda expression.

Let's see a scenario. If we don't implement Java lambda expression. Here, we are implementing an interface method without using lambda expression.

public interface FunctionalInterface

{

public void show();

}

public class TestFunInter {

public static void main(String[] args){

FunctionalInterface f=()->System.out.println("hello world");

f.show();

}

}

Example 2:

import java.util.function.\*;

public class Test{

public static void main(String[] args){

BiConsumer<String, String> bi = (a, b) -> System.out.println(a + b);

bi.accept("shiv ", "----------");

}

}

Example 3:

public class TestClass {

public static void main(String[] args){

Runnable r = () -> System.out.println("lambda expression");

Thread t = new Thread(r);

t.start();

}

}