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

1. (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.

Without Lambda Expression

1. **interface** Drawable{
2. **public** **void** draw();
3. }
4. **public** **class** LambdaExpressionExample {
5. **public** **static** **void** main(String[] args) {
6. **int** width=10;
8. //without lambda, Drawable implementation using anonymous class
9. Drawable d=**new** Drawable(){
10. **public** **void** draw(){System.out.println("Drawing "+width);}
11. };
12. d.draw();
13. }
14. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample)

Output:

Drawing 10

Java Lambda Expression Example

Now, we are implementing the above example with the help of lambda expression.

1. @FunctionalInterface  //It is optional
2. **interface** Drawable{
3. **public** **void** draw();
4. }
6. **public** **class** LambdaExpressionExample2 {
7. **public** **static** **void** main(String[] args) {
8. **int** width=10;
10. //with lambda
11. Drawable d2=()->{
12. System.out.println("Drawing "+width);
13. };
14. d2.draw();
15. }
16. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample2)

Output:

Drawing 10

A lambda expression can have zero or any number of arguments. Let's see the examples:

Java Lambda Expression Example: No Parameter

1. **interface** Sayable{
2. **public** String say();
3. }
4. **public** **class** LambdaExpressionExample3{
5. **public** **static** **void** main(String[] args) {
6. Sayable s=()->{
7. **return** "I have nothing to say.";
8. };
9. System.out.println(s.say());
10. }
11. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample3)

Output:

I have nothing to say.

Java Lambda Expression Example: Single Parameter

1. **interface** Sayable{
2. **public** String say(String name);
3. }
5. **public** **class** LambdaExpressionExample4{
6. **public** **static** **void** main(String[] args) {
8. // Lambda expression with single parameter.
9. Sayable s1=(name)->{
10. **return** "Hello, "+name;
11. };
12. System.out.println(s1.say("Sonoo"));
14. // You can omit function parentheses
15. Sayable s2= name ->{
16. **return** "Hello, "+name;
17. };
18. System.out.println(s2.say("Sonoo"));
19. }
20. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample4)

Output:

Hello, Sonoo

Hello, Sonoo

Java Lambda Expression Example: Multiple Parameters

1. **interface** Addable{
2. **int** add(**int** a,**int** b);
3. }
5. **public** **class** LambdaExpressionExample5{
6. **public** **static** **void** main(String[] args) {
8. // Multiple parameters in lambda expression
9. Addable ad1=(a,b)->(a+b);
10. System.out.println(ad1.add(10,20));
12. // Multiple parameters with data type in lambda expression
13. Addable ad2=(**int** a,**int** b)->(a+b);
14. System.out.println(ad2.add(100,200));
15. }
16. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample5)

Output:

30

300

Java Lambda Expression Example: with or without return keyword

In Java lambda expression, if there is only one statement, you may or may not use return keyword. You must use return keyword when lambda expression contains multiple statements.

1. **interface** Addable{
2. **int** add(**int** a,**int** b);
3. }
5. **public** **class** LambdaExpressionExample6 {
6. **public** **static** **void** main(String[] args) {
8. // Lambda expression without return keyword.
9. Addable ad1=(a,b)->(a+b);
10. System.out.println(ad1.add(10,20));
12. // Lambda expression with return keyword.
13. Addable ad2=(**int** a,**int** b)->{
14. **return** (a+b);
15. };
16. System.out.println(ad2.add(100,200));
17. }
18. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample6)

Output:

30

300

Java Lambda Expression Example: Foreach Loop

1. **import** java.util.\*;
2. **public** **class** LambdaExpressionExample7{
3. **public** **static** **void** main(String[] args) {
5. List<String> list=**new** ArrayList<String>();
6. list.add("ankit");
7. list.add("mayank");
8. list.add("irfan");
9. list.add("jai");
11. list.forEach(
12. (n)->System.out.println(n)
13. );
14. }
15. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample7)

Output:

ankit

mayank

irfan

jai

Java Lambda Expression Example: Multiple Statements

1. @FunctionalInterface
2. **interface** Sayable{
3. String say(String message);
4. }
6. **public** **class** LambdaExpressionExample8{
7. **public** **static** **void** main(String[] args) {
9. // You can pass multiple statements in lambda expression
10. Sayable person = (message)-> {
11. String str1 = "I would like to say, ";
12. String str2 = str1 + message;
13. **return** str2;
14. };
15. System.out.println(person.say("time is precious."));
16. }
17. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample8)

Output:

I would like to say, time is precious.

Java Lambda Expression Example: Creating Thread

You can use lambda expression to run thread. In the following example, we are implementing run method by using lambda expression.

1. **public** **class** LambdaExpressionExample9{
2. **public** **static** **void** main(String[] args) {
4. //Thread Example without lambda
5. Runnable r1=**new** Runnable(){
6. **public** **void** run(){
7. System.out.println("Thread1 is running...");
8. }
9. };
10. Thread t1=**new** Thread(r1);
11. t1.start();
12. //Thread Example with lambda
13. Runnable r2=()->{
14. System.out.println("Thread2 is running...");
15. };
16. Thread t2=**new** Thread(r2);
17. t2.start();
18. }
19. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample9)

Output:

Thread1 is running...

Thread2 is running...

Java lambda expression can be used in the collection framework. It provides efficient and concise way to iterate, filter and fetch data. Following are some lambda and collection examples provided.

Java Lambda Expression Example: Comparator

1. **import** java.util.ArrayList;
2. **import** java.util.Collections;
3. **import** java.util.List;
4. **class** Product{
5. **int** id;
6. String name;
7. **float** price;
8. **public** Product(**int** id, String name, **float** price) {
9. **super**();
10. **this**.id = id;
11. **this**.name = name;
12. **this**.price = price;
13. }
14. }
15. **public** **class** LambdaExpressionExample10{
16. **public** **static** **void** main(String[] args) {
17. List<Product> list=**new** ArrayList<Product>();
19. //Adding Products
20. list.add(**new** Product(1,"HP Laptop",25000f));
21. list.add(**new** Product(3,"Keyboard",300f));
22. list.add(**new** Product(2,"Dell Mouse",150f));
24. System.out.println("Sorting on the basis of name...");
26. // implementing lambda expression
27. Collections.sort(list,(p1,p2)->{
28. **return** p1.name.compareTo(p2.name);
29. });
30. **for**(Product p:list){
31. System.out.println(p.id+" "+p.name+" "+p.price);
32. }
34. }
35. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample10)

Output:

Sorting on the basis of name...

2 Dell Mouse 150.0

1 HP Laptop 25000.0

3 Keyboard 300.0

Java Lambda Expression Example: Filter Collection Data

1. **import** java.util.ArrayList;
2. **import** java.util.List;
3. **import** java.util.stream.Stream;
4. **class** Product{
5. **int** id;
6. String name;
7. **float** price;
8. **public** Product(**int** id, String name, **float** price) {
9. **super**();
10. **this**.id = id;
11. **this**.name = name;
12. **this**.price = price;
13. }
14. }
15. **public** **class** LambdaExpressionExample11{
16. **public** **static** **void** main(String[] args) {
17. List<Product> list=**new** ArrayList<Product>();
18. list.add(**new** Product(1,"Samsung A5",17000f));
19. list.add(**new** Product(3,"Iphone 6S",65000f));
20. list.add(**new** Product(2,"Sony Xperia",25000f));
21. list.add(**new** Product(4,"Nokia Lumia",15000f));
22. list.add(**new** Product(5,"Redmi4 ",26000f));
23. list.add(**new** Product(6,"Lenevo Vibe",19000f));
25. // using lambda to filter data
26. Stream<Product> filtered\_data = list.stream().filter(p -> p.price > 20000);
28. // using lambda to iterate through collection
29. filtered\_data.forEach(
30. product -> System.out.println(product.name+": "+product.price)
31. );
32. }
33. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=LambdaExpressionExample11)

Output:

Iphone 6S: 65000.0

Sony Xperia: 25000.0

Redmi4 : 26000.0

Java Lambda Expression Example: Event Listener

1. **import** javax.swing.JButton;
2. **import** javax.swing.JFrame;
3. **import** javax.swing.JTextField;
4. **public** **class** LambdaEventListenerExample {
5. **public** **static** **void** main(String[] args) {
6. JTextField tf=**new** JTextField();
7. tf.setBounds(50, 50,150,20);
8. JButton b=**new** JButton("click");
9. b.setBounds(80,100,70,30);
11. // lambda expression implementing here.
12. b.addActionListener(e-> {tf.setText("hello swing");});
14. JFrame f=**new** JFrame();
15. f.add(tf);f.add(b);
16. f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
17. f.setLayout(**null**);
18. f.setSize(300, 200);
19. f.setVisible(**true**);
21. }
23. }

Output: