Lambda Expressions Java 8 Example

Lambda expressions are one of the important features included in java 8.

Lambda Expressions are used to write concise code for that interface which consist one method i.e. *functional interface*.

Lambda Expression is also used to iterate collection in simple way. For a single method interface whatever we achieve by using anonymous class or inner class, the same can be achieved using lambda expression in a very concise code in java 8.

Lambda expression defines the functional interface method and returns the instance of that interface.

Lambda Expression Syntax

Lambda Expression Syntax is very easy to read and understand. Lambda Expression Syntax will look like 

(Argument part) -> Body part

Find some example below.   
**Sample 1:** If method takes no argument and printing the message. 

() -> System.out.println("Your message");

**Sample 2 :** If method takes two arguments and do some business logic and returns the value. 

(int a, int b) -> a+b;

The value of a+b will be returned by the method.   
  
**Sample 3:** If method takes one argument and do some business logic 

(String s) -> s + "Hello World";

String will be returned after concatenation.

Environment Setup to Run Lambda Expression Examples

Now we will run some example to understand lambda expression. To run the example we need below software.   
1. JDK 8   
2. NetBeans or Eclipse supporting Java 8. To run the demo I am using NetBeans IDE 8.0

Example 1: Using Runnable with Lambda Expression

In the below example we are running a Runnable thread. Before lambda expression, to achieve runnable thread, we need to define a class which implements Runnable interface to get runnable object. Now see how to achieve using lambda expressions the same object.

**RunnableDemo.java**

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class RunnableDemo {

public static void main(String[] args) {

final ExecutorService exService = Executors.newSingleThreadExecutor();

Runnable r = () -> System.out.println("Lambda Expression Test with Runnable");

exService.execute(r);

}

}

The output will be 

Lambda Expression Test with Runnable

Example 2: User defined Functional Interface Example with Lambda Expression

In this example we will create our own functional interface and will see how to use it with lambda expression. Find functional interface.   
**Calculator.java**

public interface Calculator {

public int add(int n1, int n2);

}

Find the main class how to use the functional interface using lambda expression.   
**CalculatorDemo**

public class CalculatorDemo {

public static void main(String[] args) {

Calculator cal =(int a, int b) -> a+b;

int res = cal.add(5, 6);

System.out.println(res);

}

}

The output will be 11.

Example 3: Using Comparator with Lambda Expression

Now see Comparator interface how to use with lambda expression to sort the collection containing user defined objects.   
**ComparatorDemo.java**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class ComparatorDemo {

public static void main(String[] args) {

List<Student> list = new ArrayList();

list.add(new Student("Ram",20));

list.add(new Student("Shyam",22));

list.add(new Student("Kabir",18));

System.out.println("...befor sorting...");

for(Student s : list){

System.out.println(s.getName());

}

//define comparator

Comparator<Student> comp= (Student s1, Student s2) -> s1.getName().compareTo(s2.getName());

Collections.sort(list, comp);

System.out.println("...after sorting...");

for(Student s : list){

System.out.println(s.getName());

}

}

}

Find the Student class with some attribute defined.   
**Student.java**

public class Student {

private String name;

private int age;

public Student(String name,int age){

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

The output will be

...befor sorting...

Ram

Shyam

Kabir

...after sorting...

Kabir

Ram

Shyam

Example 4: Using Function Interface to Iterate Collection with Lambda Expression

In java 8 there is *java.util.function* package introduced. With the help of java.util.function.Function API, we will iterate collection using lambda expression. In Function interface there is *apply()*method that will be called in user defined function. Suppose we have to create a method for custom print then we will define the method as below.

public String customShow(Function<Student,String> fun){

return fun.apply(this);

}

Add the above method in Student class to run the below example.   
**FunctionDemo.java**

import java.util.ArrayList;

import java.util.List;

public class FunctionDemo {

public static void main(String[] args) {

List<Student> list = new ArrayList();

list.add(new Student("Ram",20));

list.add(new Student("Shyam",22));

list.add(new Student("Kabir",18));

for(Student st: list){

System.out.println(st.customShow(s->s.getName()+": "+s.getAge()));

}

}

}

The output will be 

Ram: 20

Shyam: 22

Kabir: 18