# Lambda Expressions | Assignment 8

------------------------------------------------------------------------------

1. Write an application to perform basic arithmetic operations like add,

subtract, multiply & divide. You need to define a functional interface first.

```sh

import java.util.Scanner;

@FunctionalInterface

interface Operator<T> {

T process(T a, T b);

}

public class LambdaOperations {

public static void main(String[] args) {

// TODO Auto-generated method stub

Operator<Integer> addOperation = (a, b) -> a + b;

System.out.println("addOperation : "+addOperation.process(10, 5));

Operator<Integer> subtractOperation = (a, b) -> a - b;

System.out.println("subtractOperation : "+subtractOperation.process(10, 5));

Operator<Integer> multiplyOperation = (a, b) -> a \* b;

System.out.println("multiplyOperation : "+multiplyOperation.process(10, 5));

Operator<Integer> divideOperation = (a, b) -> a / b;

System.out.println("divideOperation : "+divideOperation.process(10, 5));

}

}

```

#### Output

```sh

addOperation : 15

subtractOperation : 5

multiplyOperation : 50

divideOperation : 2

```

2. Write an application using lambda expressions to print Orders having 2

criteria implemented: 1) order price more than 10000 2) order status is

ACCEPTED or COMPLETED.

##### Orderclass

```sh

public class Orderclass {

int price;

String status;

public Orderclass(int price, String status) {

super();

this.price = price;

this.status = status;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

@Override

public String toString() {

return "Orderclass [price=" + price + ", status=" + status + "]";

}

}

```

#### Mainclass(Orders)

```sh

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

import java.util.function.Predicate;

public class Orders {

public static void main(String[] args) {

// TODO Auto-generated method stub

List<Orderclass> orderdetails =Arrays.asList(

new Orderclass(15000,"ACCEPTED"),

new Orderclass(9000,"ACCEPTED"),

new Orderclass(20000,"COMPLETED"),

new Orderclass(18000,"COMPLETED"),

new Orderclass(5000,"ACCEPTED")

);

//printing orders price more than 10000 and their status

System.out.println("--------Printing Orders----------");

performconditionaly(orderdetails,o->o.getPrice()>10000,o->System.out.println("Order Price : "+o.getPrice()+" Order Status : "+o.getStatus()));

}

private static void performconditionaly(List<Orderclass> orderdetails,Predicate<Orderclass> predicate,Consumer<Orderclass> consumer) {

// TODO Auto-generated method stub

for(Orderclass o: orderdetails) {

if(predicate.test(o)) {

consumer.accept(o);

}

}

}

}

```

#### Output

```sh

--------Printing Orders----------

Order Price : 15000 Order Status : ACCEPTED

Order Price : 20000 Order Status : COMPLETED

Order Price : 18000 Order Status : COMPLETED

```

1. Use the functional interfaces Supplier, Consumer, Predicate & Function to

invoke built-in methods from Java API.

```sh

import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class LambdaInterfaces {

public static void main(String[] args) {

//consumer functional interface

String str = "Consumer Interface";

Consumer<String>displayConsumer = a->System.out.println(a);

displayConsumer.accept(str.toUpperCase());

//Predicate functional interface

Predicate<String> displaypredicate= p ->str.length() > 10;

System.out.println("Predicate functional interface: "+displaypredicate.test(str));

//Function functional interface

Function<Integer,Double>val = a ->a / 5.0;

System.out.println("Function functional interface: "+val.apply(37));

//Supplier functional interface

Supplier<Float>suppval = () ->Math.max(18.99f, 19.9f);

System.out.println("Supplier functional interface: "+suppval.get());

}

}

```

#### Output

```sh

CONSUMER INTERFACE

Predicate functional interface: true

Function functional interface: 7.4

Supplier functional interface: 19.9

```

4. Remove the words that have odd lengths from the list. HINT: Use one of the

new methods from JDK 8. Use removelf() method from Collection interface.

```sh

import java.util.ArrayList;

public class Oddlength {

public static void main(String[] args) {

// TODO Auto-generated method stub

ArrayList<String> words =new ArrayList<String>();

words.add("Hello");

words.add("Welcome");

words.add("Computer");

words.add("System");

words.add("Assignment");

words.add("Collections");

words.removeIf(w->(w.length()%2==0));

words.forEach(System.out::println);

}

}

```

#### Output

```sh

Hello

Welcome

Collections

```

5.Create a string that consists of the first letter of each word in the list of Strings provided. HINT: Use Consumer interface & a StringBuilder to

construct the result.

```sh

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

public class AppendResult {

public static void main(String[] args) {

// TODO Auto-generated method stub

StringBuilder str=new StringBuilder();

List<String> names =Arrays.asList(

new String("What"), new String("Else"),

new String("Look"), new String("Like"),

new String("Dust"), new String("Or"),

new String("Not"),new String("Edible")

);

for(String n : names) {

str.append(n.charAt(0));

}

//prints the first letters of all the string in the list

printstring(str,c->System.out.println(str));

}

private static void printstring(StringBuilder str,Consumer consumer) {

// TODO Auto-generated method stub

if(str!=null) {

consumer.accept(str);

}

}

}

```

#### Output

```sh

WELLDONE

```

6.Replace every word in the list with its upper case equivalent. Use

replaceAll() method & UnaryOperator interface.

```sh

import java.util.Arrays;

import java.util.List;

import java.util.function.UnaryOperator;

class replace implements UnaryOperator<String>{

public String apply(String str) {

return str.toUpperCase();

}

}

public class Unaryopt {

public static void main(String[] args) {

// TODO Auto-generated method stub

List<String> names =Arrays.asList(

new String("What"), new String("Else"),

new String("Look"), new String("Like"),

new String("Dust"), new String("Or"),

new String("Not"),new String("Edible")

);

System.out.println("list before replace operation: "+names);

names.replaceAll(new replace());

System.out.println("Contents of the list after replace operation: \n"+names);

}

}

```

##### Output

```sh

list before replace operation: [What, Else, Look, Like, Dust, Or, Not, Edible]

Contents of the list after replace operation:

[WHAT, ELSE, LOOK, LIKE, DUST, OR, NOT, EDIBLE]

```

7.Convert every key-value pair of the map into a string and append them all

into a single string, in iteration order. HINT: Use Map.entrySet() method & a

StringBuilder to construct the result String.

```sh

import java.util.Map;

import java.util.TreeMap;

public class Convert {

public static void main(String[] args) {

// TODO Auto-generated method stub

StringBuilder str=new StringBuilder();

Map<Integer,String> map = new TreeMap<>();

map.put(1, "Hello");

map.put(2, "Guys");

map.put(3, "How");

map.put(4, "Are");

map.put(5, "You");

map.put(6, "Doing");

map.put(7, "In");

map.put(8, "Training");

for(Map.Entry<Integer,String>entry:map.entrySet()) {

Integer key = entry.getKey();

String c = entry.getValue();

str.append(key + c);

}

//print result string

System.out.println(str);

}

}

```

#### Output

```sh

1Hello2Guys3How4Are5You6Doing7In8Training

```

8.Create a new thread that prints the numbers from the list. Use class Thread

& interface Consumer.

```sh

import java.util.ArrayList;

import java.util.List;

public class Threadlist {

public static void main(String[] args) {

// TODO Auto-generated method stub

List<Integer> num=new ArrayList<Integer>(){{

add(11);

add(55);

add(37);

add(95);

add(99);

}};

Thread mylambda = new Thread(()->System.out.println(num));

mylambda.run();

}

}

```

#### Output

```sh

[11, 55, 37, 95, 99]

```