Lambda Expressions

1:-

package lambdaAssignments;

interface Arithmetic{

int operation(int a,int b);

}

public class Operations {

public static void main(String[] args) {

Arithmetic Addition=(int a,int b)-> (a+b);

System.out.println("Addition:"+ Addition.operation(4,6));

Arithmetic Subtraction=(int a,int b)-> (a-b);

System.out.println("Subtraction:"+ Subtraction.operation(4,6));

Arithmetic Multiplication=(int a,int b)-> (a\*b);

System.out.println("Multiplication:"+ Multiplication.operation(4,6));

Arithmetic Division=(int a,int b)-> (a/b);

System.out.println("Division:"+ Division.operation(4,6));

}

}

2:-

Main class:-

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

public class OrderMain {

public static void main(String[] args) {

List<Order> or=new ArrayList<>(Arrays.asList(

new Order(1,5000,"Accepted"),

new Order(2,15000,"Completed"),

new Order(3,26000,"Pending"),

new Order(4,10000,"Accepted"),

new Order(5,32000,"Accepted")

));

for(Order o:or)

System.out.println(o);

System.out.println("Orders having two criteria");

or.stream()

.forEach(i -> {

if(i.getPrice()>10000 && (i.getStatus()=="Accepted" || i.getStatus()=="Completed"))

System.out.println(i);

});

}

}

Order class:-

public class Order {

int id;

int price;

String status;

public Order(int id, int price, String status) {

super();

this.id = id;

this.price = price;

this.status = status;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

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 "Order [id=" + id + ", price=" + price + ", status=" + status + "]";

}

}

3:-

Supplier

import java.util.function.Supplier;

public class SupplierExample {

static String product = "Android";

public static void main(String[] args) {

Supplier<Boolean> boolSupplier = () -> product.length() == 10;

Supplier<Integer> intSupplier = () -> product.length() - 2;

Supplier<String> supplier = () -> product.toUpperCase();

System.out.println(boolSupplier.get());

System.out.println(intSupplier.get());

System.out.println(supplier.get());

}

}

Main:-

import java.util.Arrays;

import java.util.List;

import java.util.function.Predicate;

public class PredicateExample {

public static void main(String args[])

{

List<String> names = Arrays.asList("Java","JavaCore","Python","JavaString","CSharp");

Predicate<String> p = (s)->s.startsWith("J");

for (String st:names)

{

if (p.test(st))

System.out.println(st);

}

}

}

Consumer

import java.util.function.Consumer;

public class ConsumerExample {

public static void main(String args[])

{

Consumer<Integer> display = a -> System.out.println(a);

display.accept(10);

}

}

Consumer main:

import java.util.function.Function;

public class FunctionExample {

static String show(String message){

return "Hello "+message;

}

public static void main(String[] args) {

Function<String, String> fun = FunctionExample::show;

System.out.println(fun.apply("Peter"));

}

}

4:-

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class RemoveOdd {

public static void main(String[] args) {

List<Integer> numbers = new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8,9,10));

numbers.removeIf( number -> number%2 != 0 );

System.out.println(numbers);

}

}

5:-

package lambdaAssignments;

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

public class FirstLetter {

public static void main(String args[])

{

List<String> strings = Arrays.asList("Violet", "Blue", "Yellow", "Red", "White");

StringBuilder sb = new StringBuilder();

Consumer<String> consumer = c -> sb.append(c.charAt(0));

strings.forEach(s -> consumer.accept(s));

System.out.println("String: " + sb);

}

}

6:-

import java.util.ArrayList;

import java.util.function.UnaryOperator;

class Op implements UnaryOperator<String> {

public String apply(String str) {

return str.toUpperCase();

}

}

public class Test {

public static void main(String[] args) throws CloneNotSupportedException {

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

list.add("Java");

list.add("JavaScript");

System.out.println("Contents of the list: "+list);

list.replaceAll(new Op());

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

}

}

7:-

**package** lambdaAssignments;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.function.BiConsumer;

**public** **class** Convert {

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

Map<Integer, String> color = **new** HashMap<Integer, String>();

color.put(1, "Yellow");

color.put(2, "Red");

color.put(3, "Blue");

color.put(4, "Black");

color.put(5, "White");

StringBuilder sb = **new** StringBuilder();

BiConsumer<Integer,String> consumer = (key, value) -> sb.append(key + value);

color.entrySet().forEach(c -> consumer.accept(c.getKey(), c.getValue()));

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

}

}

8:-

**package** lambdaAssignments;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** PrintNumbers {

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

List<Integer> num = **new** ArrayList<>(Arrays.*asList*(1,2,3,4,5));

Thread thread = **new** Thread(() -> num.forEach(n -> System.***out***.println(n)));

thread.run();

}

}