**Lambda expression assignments**

1. Write an application to perform basic arithmetic operations like add, substract, multiply & divide. You need to define a functional interface first.

**interface** Arithmetic {

**int** operation(**int** x, **int** y);

}

**public** **class** Functionalinterface {

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

// **TODO** Auto-generated method stub

// Addition using Lambda expression

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

// Arithmetic addition = (int a, int b) -> {return a + b;};

System.***out***.println("Addition = " + addition.operation(17, 12));

// Subtraction using Lambda expression

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

// Arithmetic addition = (int a, int b) -> {return a - b;};

System.***out***.println("Subtraction = " + subtraction.operation (51,23));

// Multiplication using Lambda expression

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

// Arithmetic addition = (int a, int b) -> {return a \* b;};

System.***out***.println("Multiplication = " + multiplication.operation (13, 24));

// Division using Lambda expression

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

// Arithmetic addition = (int a, int b) -> {return a \* b;};

System.***out***.println("Division = " + division.operation(50, 5));

}

}

**Output:**  Addition = 29

Subtraction = 28

Multiplication = 312

Division = 10

1. 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.

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.List;

**import** java.util.function.\*;

**import** java.util.stream.Stream;

**class** order {

**int** orderId;

Double cost;

String orderStatus;

**public** order(**int** orderId, Double cost, String orderStatus) {

**super**();

**this**.orderId = orderId;

**this**.cost = cost;

**this**.orderStatus = orderStatus;

}

**public** **int** getOrderId() {

**return** orderId;

}

**public** **void** setOrderId(**int** orderId) {

**this**.orderId = orderId;

}

**public** Double getCost() {

**return** cost;

}

**public** **void** setCost(Double cost) {

**this**.cost = cost;

}

**public** String getOrderStatus() {

**return** orderStatus;

}

**public** **void** setOrderStatus(String orderStatus) {

**this**.orderStatus = orderStatus;

}

@Override

**public** String toString() {

// **TODO** Auto-generated method stub

//return super.toString();

**return** "order [orderId=" + orderId + ", cost=" + cost + ", orderStatus=" + orderStatus + "]";

}

}

**mport** java.util.ArrayList;

**import** java.util.List;

**import** java.util.stream.Stream;

**public** **class** Price {

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

// **TODO** Auto-generated method stub

List<order> list=**new** ArrayList<order>();

list.add(**new** order(1,2500.0,"Accepted"));

list.add(**new** order(2,10000.0,"Completed"));

list.add(**new** order(3,15000.0,"Accepted"));

list.add(**new** order(4,9500.0,"Completed"));

//printing all the order price more than 10000 and printing it's status

Stream<order> filtered\_data = list.stream().filter(o -> o.cost >= 10000);

filtered\_data.forEach(

order -> System.***out***.println(order.cost+": "+order.orderStatus));

}

}

**Output:** 10000.0: Completed

15000.0: Accepted

1. Use the functional interfaces Supplier, Consumer, Predicate & Function to invoke built-in methods from Java API.

**class** product {

**private** **double** price = 0.0;

**public** product(**double** price) {

**super**();

**this**.price = price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** **void** printprice() {

System.***out***.println("This is Consumer functional interface: " +price);

}

}

**import** java.util.function.Consumer;

**import** java.util.function.Function;

**import** java.util.function.Predicate;

**import** java.util.function.Supplier;

**public** **class** product1 {

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

// **TODO** Auto-generated method stub

//consumer functional interface

Consumer<product>updatePrice = p ->p.setPrice(9.7);

product p = **new** product(2.0);

updatePrice.accept(p);

p.printprice();

//Predicate functional interface

Predicate<String>isALongWord = t ->t.length() > 10;

String s = "successfully";

**boolean** result = isALongWord.test(s);

System.***out***.println("This is Predicate functional interface: "+s);

//Function functional interface

Function<Integer, Double>half = a ->a / 2.0;

System.***out***.println("This is Function functional interface: "+half.apply(10));

//Supplier functional interface

Supplier<Double>randomValue = () ->Math.*random*();

System.***out***.println("This is Supplier functional interface: "+randomValue.get());

}

}

**Output:** This is Consumer functional interface: 9.7

This is Predicate functional interface: successfully

This is Function functional interface: 5.0

This is Supplier functional interface: 0.34524593974962003

1. Remove the words that have odd lengths from list. **HINT:** Use one of the new methods from JDK 8. Use removeIf() method from Collection Interface.

**import** java.util.\*;

**public** **class** words

{

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

// **TODO** Auto-generated method stub

List<String> word = **new** ArrayList<String>();

word.add("Mobile");

word.add("Television");

word.add("Laptop");

word.add("Window");

word.add("Glass");

word.add("Chair");

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

word.forEach(System.***out***::println);

}

}

**Output:** Mobile

Television

Laptop

Window

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

**import** java.util.List;

**import** java.util.function.Consumer;

**public** **class** word {

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

// **TODO** Auto-generated method stub

**var** word = List.*of*("Apple", "Ball", "Cat", "Dog", "Egg");

word.forEach(**new** Consumer<String>()

{

**public** **void** accept (String s)

{

StringBuilder s1 = **new** StringBuilder();

**for** (String st : s.split(" "))

{

s1.append(st.charAt(0));

}

System.***out***.println(s1.toString());

}

});

}}

**Output:** A

B

C

D

E

1. Replace every word in the list with its upper case equivalent. Use replaceAll() method & UnaryOperator interface.

**import** java.util.ArrayList;

**import** java.util.function.UnaryOperator;

**class** UpperCaseConversion **implements** UnaryOperator<String>{

**public** String apply(String str) {

**return** str.toUpperCase();

}

}

**public** **class** uppercase {

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

ArrayList<String> l = **new** ArrayList<>();

l.add("rob");

l.add("sam");

l.add("robert");

l.add("emma");

l.add("daniel");

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

//replace all elements to uppercase

l.replaceAll(**new** UpperCaseConversion());

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

}

}

**Output:** Contents of the list: [rob, sam, robert, emma, daniel]

Contents of the list after replace operation:[ROB, SAM, ROBERT, EMMA, DANIEL]

1. 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.

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.stream.Collectors;

**public** **class** maptostring {

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

// **TODO** Auto-generated method stub

Map<String, String> map = **new** HashMap<>(5);

map.put("J.K.", "Rowling");

map.put("Mark", "Twain");

map.put("Charles", "Dickens");

map.put("Lewis", "Carroll");

map.put("William", "Shakespeare");

String s = map.entrySet().stream().map((entry) ->

" " + entry.getKey() + " " + entry.getValue().replaceAll("\s ", "\\s ") + " ")

.collect(Collectors.*joining*(" "));

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

}

}

**Output:** Mark Twain Charles Dickens William Shakespeare J.K. Rowling Lewis Carroll

1. Create a new thread that prints the numbers from the list. Use class Thread & interface Consumer.

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Num {

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

// **TODO** Auto-generated method stub

List<Integer> l = **new** ArrayList<Integer>(){{

add(23);

add(98);

add(43);

add(13);

add(64);

}};

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

mylambda.run();

}

}

**Output:** [23, 98, 43, 13, 64]