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

Interface: Arithmetic

package lambda;

public interface Arithmetic

{

int operation(int a, int b);

}

Class : ArithemticOperation

package lambda;

public class AirthmeticOperations

{

public static void main(String[] args)

{

// 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(1,1));

// 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(1,1));

// 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(1,1));

// 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(4,2));

}

}

OUTPUT:

Addition = 2

Subtraction = 0

Multiplication = 1

Division = 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

Interface: Price

package lambda;

public interface Price

{

public String price();

}

Class : FunctionalInterface

package org.lambda.app;

public class FunctionalInterface

{

public static void main(String[] args) {

Integer i = 200000;

Price p = () -> {

if (i > 10000)

{

return "Accepted!";

} else

return "Not Accepted!";

};

System.out.println(p.price());

}

}

OUTPUT :

Accepted!

3. Use the functional interfaces Supplier, consumer, Predicate and Function to invoke built-in

Methods from java API.

Class : Product.java

package lambda;

import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;

class Product

{

private double price = 0.0;

public void setPrice(double price)

{

this.price = price;

}

public void printPrice() {

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

}

}

Class : ProductMainmethod.java

package lambda;

import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class ProductMainmethod

{

public static void main(String[] args)

{

//consumer functional interface

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

Product p = new Product();

updatePrice.accept(p);

p.printPrice();

//Predicate functional interface

Predicate<String>isALongWord = t ->t.length() > 12;

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 / 4.0;

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: 2.5

This is supplier functional interface: 0.6368305434925969

4.Remove the words that have odd lengths from the list. HINT : Use one of the new methods from JDK 8. Use remove() method from Collection interface.

package lambda;

import java.util.\*;

public class Remove

{

public static void main(String args[]){

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

words.add("Swapna");

words.add("Mounika");

words.add("Srilekha");

words.add("Harika");

words.add("Manasa");

words.add("Srija");

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

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

}

}

OUTPUT:

Mounika

Harika

Srija

5. Create a string that consists of the first letter of each word in the list of Strings provide. HINT : Use Consumer interface and a StringBulider to construct the results.

package lambda;

import java.util.List;

import java.util.function.Consumer;

public class StringBulider {

public static void main(String[] args) {

var word = List.of("Cherry", "Mongo", "Apple", "Grapes");

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:

C

M

A

G

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

Class : UpperCaseConverstion.java

package lamda;

import java.util.ArrayList;

import java.util.function.UnaryOperator;

class UpperCaseConverstion implements UnaryOperator<String> {

public String apply(String str) {

return str.toUpperCase();

}

}

Class : UppercaseMainMethod.java

package lambda;

import java.util.ArrayList;

public class UppercaseMainMethod {

public static void main(String[] args) {

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

list.add(github);

list.add("mongodb");

list.add("java");

list.add("python");

list.add("testing");

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

list.replaceAll(new UpperCaseConverstion());

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

}

}

OUTPU

Contents of the list after replace operation:[GITHUB,MONGODB,JAVA, PYTHON, TESTING]

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 StringBulider to construct the result String.

package org.lambda.app;

import java.util.HashMap;

import java.util.Map;

import java.util.stream.Collectors;

public class MapToString {

public static void main(String[] cmd\_lineParams) {

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

map.put("Swapna", "1");

map.put("Srarika", "5");

map.put("Kavya", "3");

map.put("Manasa", "4");

map.put("pranaya", "2");

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

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

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

System.out.println(s);

}

}

OUTPUT:

Swapna "1" Sarika "5" Kavya "3" Manasa "4" Pranaya "2"

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

package org.lambda.app;

import java.util.ArrayList;

import java.util.List;

public class ListToString {

public static void main(String[] args)

{

List<Integer> n=new ArrayList<Integer>()

{{

add(1);

add(4);

add(5);

add(10);

add(11);

} };

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

mylambda.run();

}

}

OUTPUT:

[1,4,5,10,11]