**Assignment 8**

**1]ANS:-**

public class AirthmeticOperations

{

public static void main(String[] args)

{

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

System.out.println("Addition = " + addition.operation(5, 6));

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

System.out.println("Subtraction = " + subtraction.operation(5, 3));

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(12, 6));

}

}

public interface Arithmetic

{

int operation(int a, int b);

}

**OUTPUT:-**

Addition = 11

Subtraction = 2

Multiplication = 24

Division = 2

**2]ANS:-**

public class FunctionalInterface

{

public static void main(String[] args) {

Integer i = 35000;

Price p = () -> {

if (i > 10000)

{

return " ACCEPTED";

} else

return "NOT ACCEPTED";

};

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

}

}

public interface Price

{

public String price();

}

**Output:-**

ACCEPTED

**3]ANS:-**

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);

}

}

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

Product p = new Product();

updatePrice.accept(p);

p.printPrice();

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<Integer, Double>half = a ->a / 4.0;

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

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]ANS:-**

import java.util.\*;

public class Remove

{

public static void main(String args[]){

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

words.add("Rushikesh");

words.add("Rutuja");

words.add("Gajanan");

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

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

}

}

**OUTPUT:-**

Rutuja

**5]ANS:-**

import java.util.List;

import java.util.function.Consumer;

public class StringBulider {

public static void main(String[] args) {

var word = List.of("Sachin", "Virat", "Dhoni", "Rohit");

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:-**

S

V

D

R

**6]ANS:-**

import java.util.ArrayList;

public class UppercaseMainMethod {

public static void main(String[] args) {

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

list.add("Java");

list.add("Spring");

list.add("c language");

list.add("python");

list.add("junit");

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

list.replaceAll(new UpperCaseConverstion());

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

}

}

import java.util.ArrayList;

import java.util.function.UnaryOperator;

class UpperCaseConverstion implements UnaryOperator<String> {

public String apply(String str) {

return str.toUpperCase();

}

}

**OUTPUT:-**

Contents of the list: [Java, Spring, c language, python, junit]

Contents of the list after replace operation:[JAVA, SPRING, C LANGUAGE, PYTHON, JUNIT]

**7]ANS:-**

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("Sachin", "1");

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

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

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

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

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

System.out.println(s);

}

}

**OUTPUT:-**

Sachin "1" Dhoni "3" Virat "2"

**8]ANS:-**

import java.util.ArrayList;

import java.util.List;

public class ListToString {

public static void main(String[] args)

{

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

{{

add(15);

add(65);

add(73);

add(99);

add(89);

} };

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

mylambda.run();

}

}

OUTPUT:

[15, 65, 73, 99, 89]

**9]ANS:-**