**JAVA 8 EXERCISE SHRIYA GARG**

1. Write the following a functional interface and implement it using lambda:
   1. (1) First number is greater than second number or not Parameter (int ,int ) Return boolean
   2. (2) Increment the number by 1 and return incremented value Parameter (int) Return int
   3. calccalccalc(3) Concatination of 2 string Parameter (String , String ) Return (String)
   4. (4) Convert a string to uppercase and return . Parameter (String) Return (String)

package Java8;

/\*

\* Write the following a functional interface and implement it using lambda:

(1) First number is greater than second number or not Parameter (int ,int ) Return boolean

(2) Increment the number by 1 and return incremented value Parameter (int) Return int

(3) Concatination of 2 string Parameter (String , String ) Return (String)

(4) Convert a string to uppercase and return . Parameter (String) Return (String)

\*/

@FunctionalInterface

interface Compute{

boolean computeMe(int a, int b); }

// boolean computeMee(int a, int b); }

@FunctionalInterface

interface Increment{

int incrementMe(int a);

}

@FunctionalInterface

interface Concatenate{

String concateMe(String a, String b);}

@FunctionalInterface

interface UpperCaseMe{

String uppMe(String s);

}

public class Q1FuncInterface {

public static void main(String[] args) {

Compute comp = (c, d) -> {

if (c > d) return true;

else return false;

};

System.*out*.println(comp.computeMe(15, 8));

System.*out*.println(comp.computeMe(15, 68));

System.*out*.println("====================================");

Increment incre = (c) -> {

c += 1;

System.*out*.println(c);

return c;

};

incre.incrementMe(4);

//System.out.println(incre.incrementMe(4));

System.*out*.println("====================================");

Concatenate conc = (s1, s2) -> {

System.*out*.println(s1 + s2);

return s1 + s2;

};

conc.concateMe("Shriya", "Garg");

System.*out*.println("====================================");

UpperCaseMe upperCaseMe = (s) -> {

return s.toUpperCase();

};

System.*out*.println(upperCaseMe.uppMe("Shriya"));

}

}

1. Create a functional interface whose method takes 2 integers and return one integer.

package Java8;

/\*

\* Create a functional interface whose method takes 2 integers and return one integer.

\*/

@FunctionalInterface

interface CalculateSum{

int calcMe(int c, int d);

}

public class Q2FunctionalTwo2One {

//FI Method

public static void main(String[] args) {

CalculateSum calc= (a,b)->{

System.*out*.println(a+b);

return a+b;};

calc.calcMe(3,7);

}}

1. Using (instance) Method reference create and apply add and subtract method and using (Static) Method reference create and apply multiplication method for the functional interface created.

package Java8;

//3. Using (instance) Method reference create and apply add and subtract method and using (Static) Method

// reference create and apply multiplication method for the functional interface created.

@FunctionalInterface

interface Interf1{

int interfaceMethod(int a, int b);

}

class MyClass{

int add(int a, int b){

return a+b;

}

int subtract(int a, int b)

{

return a-b;

}

static int multiply(int a, int b)

{

return a\*b;

}

}

public class Q3StaticFunctionTest {

public static void main(String[] args) {

//Method Reference class method

Interf1 interface1 = new MyClass()::add;

System.*out*.println("Adding 7 and 8 :: "+interface1.interfaceMethod(7,8));

interface1 = new MyClass()::subtract;

System.*out*.println("Subtracting 7 and 8 :: "+interface1.interfaceMethod(7, 8));

interface1=MyClass::*multiply*;

System.*out*.println("Multiplying 7 and 8 :: "+interface1.interfaceMethod(7, 8));

}

}

1. Create an Employee Class with instance variables (String) name, (Integer)age, (String)city and get the instance of the Class using constructor reference

package Java8;

/\*Q4

\* Create an Employee Class with instance variables (String) name, (Integer)age,

\* (String)city and get the instance of the Class using constructor reference

\*/

interface EmployeeHolder{

Employee constructorRefMethod(String name, Integer age, String city);

}

class Employee{

String name;

Integer age;

String city;

public Employee(String name, Integer age, String city) {

this.name = name;

this.age = age;

this.city = city;

// System.out.println("Name : " + name + ", Age : " + age + ", City : " + city);

}

@Override

public String toString() {

return ("Name : "+name+", Age : "+age+", City : "+city);

}

}

public class Q4EmployeeClass {

public static void main(String[] args) {

// Constructor Refernce

EmployeeHolder empHol = Employee::new;

System.*out*.println(empHol.constructorRefMethod("Shriya", 23, "Delhi"));

System.*out*.println(empHol.constructorRefMethod("Reema", 21, "Ahemdabad"));

}

}

1. Implement following functional interfaces from java.util.function using lambdas:

(1) Consumer

(2) Supplier

(3) Predicate

(4) Function

package Java8;

/\*

5. Implement following functional interfaces from java.util.function using lambdas:

(1) Consumer

(2) Supplier

(3) Predicate

(4) Function

\*/

import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class Q5FuncInterfacesJavaUtil {

public static void main(String[] args) {

// Predicate

Predicate<Integer> pred = num ->num>10;

System.*out*.println("Is 15 is greater than 10 ? "+pred.test(15));

// Function

Function<String, Integer> function = myString-> myString.length();

System.*out*.println("Calculating th length of String supplied i.e 'divya' "+function.apply("divya"));

// Consumer

Consumer<String> consumer = (str) -> {

str = str.concat("World");

System.*out*.println("Concatinatng the given 2 Strings we have: "+str); };

consumer.accept("Hello");

// Supplier

Supplier<Integer> supplier = ()-> 8;

System.*out*.println("Supplied Value is: "+supplier.get());

}

}

1. Create and access default and static method of an interface.

package Java8;

/\*

\* Create and access default and static method of an interface.

\* \*/

interface myInterface{

default void displayDefault(){

System.*out*.println("Hey, its Default's Display!!");

}

static void displayStatic(){

System.*out*.println("Hey, its Static's Display!!");

}

}

public class Q6AccessMethod implements myInterface{

public static void main(String[] args) {

myInterface.*displayStatic*();

// myInterface.displayDefault();

myInterface interfaceReference = new Q6AccessMethod();

interfaceReference.displayDefault();

}

}

1. Override the default method of the interface.

package Java8;

/\*

\* Override the default method of the interface.

\*/

interface DefaultDemo{

default void display(){

System.*out*.println("Hi, I'm display's body from interface");

}

}

public class Q7OverrideDefaultMethod implements DefaultDemo{

public void display(){

System.*out*.println("Hey, it's me from Class");

}

public static void main(String[] args) {

Q7OverrideDefaultMethod defOver = new Q7OverrideDefaultMethod();

defOver.display();

}

}

1. Implement multiple inheritance with default method inside interface.

package Java8;

/\*

Implement multiple inheritance with default method inside interface.

\*/

import com.sun.xml.internal.ws.api.model.wsdl.WSDLOutput;

interface Parent1{

default void myMethod(){

System.*out*.println("My Method from Parent 1");

}

}

interface Parent2{

default void myMethod(){

System.*out*.println("My Method from Parent 2");

}

}

public class Q8MultipleInhDefault implements Parent1, Parent2{

public Q8MultipleInhDefault(){

System.*out*.println("Object Instantiated");

}

@Override

public void myMethod() {

Parent2.super.myMethod();

Parent1.super.myMethod();

}

public static void main(String[] args) {

Q8MultipleInhDefault mul = new Q8MultipleInhDefault();

mul.myMethod();

}

}

1. Collect all the even numbers from an integer list.

package Java8;

/\*

\* Collect all the even numbers from an integer list.

\*/

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Q9Even {

public static void main(String[] args) {

List<Integer> integerList = Arrays.*asList*(12, 15, 45, 23, 17, 18, 14, 16);

System.*out*.println(

integerList

.stream()

.filter((e)-> e%2==0)

.collect(Collectors.*toList*())

);

}

}

1. Sum all the numbers greater than 5 in the integer list.

package Java8;

/\*

\*Sum all the numbers greater than 5 in the integer list.

\*/

import java.util.Arrays;

import java.util.List;

public class Q10SumGreaterThan {

public static void main(String[] args) {

List<Integer> integerList = Arrays.*asList*(2, 5, 13, 14, 12, 25, 25, 31);

int sum = integerList

.stream()

.filter((e)-> e>5)

.reduce(0, (a,b)-> a+b);

System.*out*.println("Required sum is : " + sum);

}

}

1. Find average of the number inside integer list after doubling it.

package Java8;

/\*Q11

\* Find average of the number inside integer list after doubling it.

\* \*/

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Q11AverageDouble {

public static void main(String[] args) {

List<Integer> intlist = Arrays.*asList*(1,2,3,4,5,6,7,7);

Double avgg =intlist

.stream()

.collect(Collectors.*averagingInt*((e)->e\*2));

System.*out*.println(avgg);

}

}

1. Find the first even number in the integer list which is greater than 3.

package Java8;

/\*Q12

\* Find the first even number in the integer list which is greater than 3.

\* \*/

import java.util.Arrays;

import java.util.List;

import java.util.Optional;

public class Q12FirstEvenGreaterThanThree {

public static void main(String[] args) {

List<Integer> intlistt = Arrays.*asList*(3,2,5,1,7 ,6,9,12,45,2);

Optional<Integer> eveG = intlistt

.stream()

.filter((e)->e%2==0)

.filter((e)->e>3)

.findFirst();

if(eveG.isPresent())

System.*out*.println("Required Integer is : " + eveG);

else

System.*out*.println("No such integer present");

}

}