* 1.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 Interface1{

boolean compare(int a,int b);

}

interface Interface2{

int increment(int number);

}

interface Interface3{

String string\_concat(String str1,String str2);

}

interface Interface4{

String convert\_uppercase(String str);

}

class functional\_interface{

public static void main(String[] args) {

//calling comapre method of interface

Interface1 I1=(a,b)->a>b;

System.*out*.println(I1.compare(4,5));

System.*out*.println(I1.compare(5,4));

System.*out*.println(I1.compare(100,101));

//calling increment method

Interface2 I2=(a)->a+1;

System.*out*.println(I2.increment(5));

System.*out*.println(I2.increment(4));

System.*out*.println(I2.increment(101));

//calling string\_concat method

Interface3 I3=(str1,str2)->str1+str2;

System.*out*.println(I3.string\_concat("Shivam","sharma"));

System.*out*.println(I3.string\_concat("ToThe","New"));

System.*out*.println(I3.string\_concat("Hello","There"));

//calling convertouppercase method

Interface4 I4=(str1)->str1.toUpperCase();

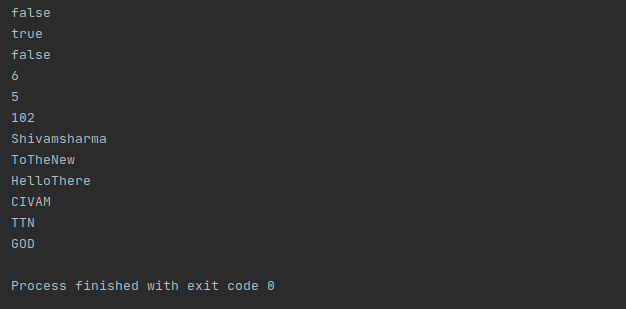
System.*out*.println(I4.convert\_uppercase("civam"));

System.*out*.println(I4.convert\_uppercase("ttn"));

System.*out*.println(I4.convert\_uppercase("god"));

}

}



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

interface fun\_Interface{

int question2(int num1,int num2);

}

public class q2\_functional\_interface{

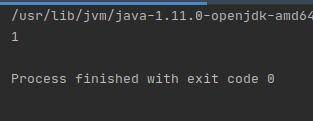
public static void main(String[] args) {

fun\_Interface fun\_interface=(n1,n2)->n1-n2;

System.*out*.println(fun\_interface.question2(3,2));

}

}



* 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.

//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.

interface GenericInterface{

void manupulation(int a, int b);

}

public class q3\_methodReference {

static void sum(int a, int b){

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

}

static void sub(int a, int b){

if(a>b)

System.*out*.println("difference is: "+(a-b));

else

System.*out*.println("difference is: "+(b-a));

}

void mult(int a, int b){

System.*out*.println("product is: "+(b\*a));

}

public static void main(String[] args) {

GenericInterface manup=q3\_methodReference::*sum*;

GenericInterface manup1=q3\_methodReference::*sub*;

GenericInterface manup2=new q3\_methodReference()::mult;

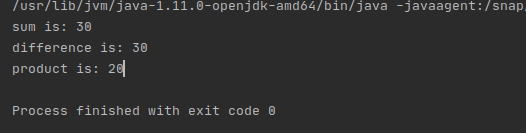
manup.manupulation(10,20);

manup1.manupulation(10,40);

manup2.manupulation(5,4);

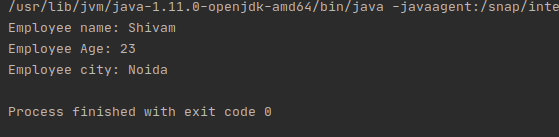
}

}



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

|  |  |
| --- | --- |
| class Employee{ |  |
|  | String name; |
|  | Integer age; |
|  | String city; |
|  | Employee(String name, Integer age, String city) { |
|  | this.name=name; |
|  | this.age=age; |
|  | this.city=city; |
|  | } |
|  | Employee getObject(){ |
|  | return this; |
|  | } |
|  | public String toString(){ |
|  | return "Employee name: "+name+"\nEmployee Age: "+age+"\nEmployee city: "+city; |
|  | } |
|  | } |
|  | interface employeeDetails{ |
|  | Employee empDetails(); |
|  | } |
|  | public class q4\_constructor\_reference  { |
|  | public static void main(String[] args) { |
|  |  |
|  | employeeDetails empdetails=new Employee("shivam", 23,"Noida")::getObject; |
|  | System.out.println(empdetails.empDetails()); |
|  |  |
|  | } |
|  | } |



* 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;

class q5\_consumerSupplier{

public static void main(String[] args) {

//Implementing Consumer interface using lambda

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

consumer.accept(10);

//Implementing Predicate interface using lambda

Predicate<Integer> predicate=i->(i>10);

System.*out*.println(predicate.test(40));

//Implementing Supplier interface using lambda

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

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

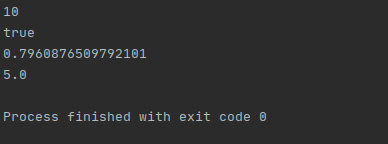
//Implementing Function interface using lambda

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

System.*out*.println(function.apply(10));

}

}



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

interface NewInterface {

static void hello()

{

System.*out*.println("Hello, New Static Method Here");

}

default void overrideMethod(String str){

System.*out*.println("Inside default method");

}

}

public class q6\_staticDefault implements NewInterface {

public static void main(String[] args)

{

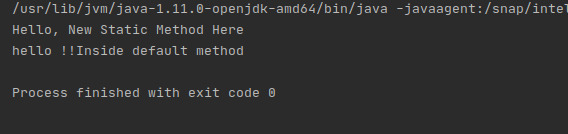
q6\_staticDefault interfaceDemo = new q6\_staticDefault();

NewInterface.*hello*();

interfaceDemo.Method2("hello !!");

}

}



* 7.Override the default method of the interface.

interface NewInterface2 {

default void overrideMethod(String str){

}

}

public class q7 implements NewInterface2 {

public static void main(String[] args) {

q7 obj = new q7();

// Calling the abstract method of interface Overriding

obj.overrideMethod("hello");

}

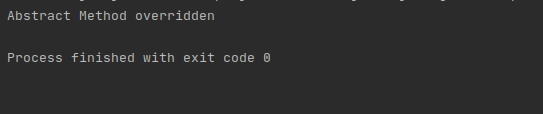
@Override

public void overrideMethod(String str1){

System.*out*.println("Abstract Method overridden");

}

}



* 8.Implement multiple inheritance with default method inside interface.

//Implement multiple inheritance with default method inside interface.

interface multInherit1{

default void show(){

System.*out*.println("from interface one");

}

}

interface multInherit2{

default void show(){

System.*out*.println("from interface two");

}

}

public class q8\_Multiface implements multInherit1, multInherit2{

@Override

public void show(){

multInherit1.super.show();

multInherit2.super.show();

}

public static void main(String[] args) {

q8\_Multiface obj=new q8\_Multiface();

obj.show();

}

}

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

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

class q9\_evennum{

public static void main(String[] args) {

List<Integer> list = Arrays.*asList*(1, 2, 3, 4,5,0,12,14);

System.*out*.println( list.stream()

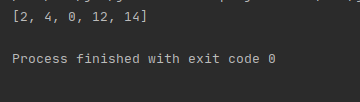
.filter(e->e%2==0)

.collect(Collectors.*toList*())

);

}

}



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

import java.util.Arrays;

import java.util.List;

class q10\_sum{

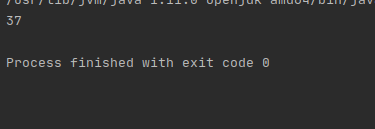
public static void main(String[] args) {

List<Integer> list = Arrays.*asList*(1, 2, 3, 4,5,0,12,14,11);

System.*out*.println(list.stream().filter(e->e>5).mapToInt(e->e).sum());

}

}



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

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

class q11\_doubleAvg{

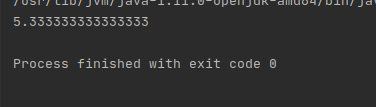
public static void main(String[] args) {

List<Integer> list = Arrays.*asList*(1,2,5);

System.*out*.println(list.stream().map(e->e\*2).collect(Collectors.*averagingDouble*(e->e)));

}

}



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

import java.util.Arrays;

import java.util.List;

class q12\_greater{

public static void main(String[] args) {

List<Integer> list = Arrays.*asList*(1,2,3,5,6,8);

System.*out*.println(list.stream().filter(e->e%2==0).filter(e->e>3).findFirst().orElse(0));

}

}

