1. Write a program to create a functional interface that has one multiplication method with 2 arguments. And method return type should be generic. And implements the functional interface method by lambda expression in another class.

Note: Your interface method should be a generic type that can accept int, float, double, and long type values.

import java.util.\*;

interface Solution<E>

{

public E multiply(E a,E b);

}

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter two number to multiply: ");

String x=sc.next();

String y=sc.next();

Solution<Integer> obj1= (Integer a, Integer b) ->

{

return a\*b;

};

Solution<Float> obj2= (Float a, Float b) ->

{

return a\*b;

};

Solution<Double> obj3= (Double a, Double b)->

{

return a\*b;

};

Solution<Long> obj4= (Long a, Long b)->

{

return a\*b;

};

try

{

System.out.println("Product of input as Double: " + obj3.multiply(Double.parseDouble(x),Double.parseDouble(y)));

System.out.println("Product of input as Float: " + obj2.multiply(Float.parseFloat(x),Float.parseFloat(y)));

System.out.println("Product of input as Long: " + obj4.multiply(Long.parseLong(x),Long.parseLong(y)));

System.out.println("Product of input as Integers: " + obj1.multiply(Integer.parseInt(x),Integer.parseInt(y)));

}

catch(NumberFormatException e)

{

System.out.println(e.toString());

}

}

}

1. Create a program to count the string words with the help of Stream API.

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

String str=sc.next()

long count=str.chars()

.filter(c -> Character.isLetter(c))

.count();

System.out.println(count);

}

}

1. Sort map on value with the help of Stream API.

import java.util.\*;

import java.util.stream.\*;

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

HashMap<String, Integer> map = new HashMap<String, Integer>();

System.out.println("Enter number of key-value pairs: ");

int n=sc.nextInt();

for(int i=0;i<n;i++)

{

System.out.println("Enter key-value pair-"+(i+1)+" :");

String key=sc.next();

int value=sc.nextInt();

map.put(key,value);

}

map.entrySet().stream()

.sorted(Map.Entry.<String, Integer>comparingByValue())

.limit(10)

.forEach(System.out::println);

}

}

1. Suppose we have a one Employee type list. Employee class have two fields name and id. Write a program to sort the list on the basis of Employee name with the help of Stream API.

import java.util.\*;

import java.util.stream.\*;

class Employee

{

String name;

int id;

Employee(String name,int id)

{

this.name=name;

this.id=id;

}

}

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

ArrayList<Employee> employee\_list=new ArrayList<Employee>();

System.out.println("Enter the number of employees: ");

int n=sc.nextInt();

String[] arr\_names=new String[n];

for(int i=0;i<n;i++)

{

System.out.println("Enter the name and Id of Employee "+(i+1));

String name=sc.next();

int id=sc.nextInt();

employee\_list.add(new Employee(name,id));

arr\_names[i]=name;

}

//DISPLAYING NAMES STARTING WITH a or A using STREAM

// List<String> list1=employee\_list.stream()

// .filter(e->e.name.charAt(0)=='a' || e.name.charAt(0)=='A')

// .map(e->e.name)

// .collect(Collectors.toList());

// System.out.println(list1);

//SORTING LIST NAMEWISE USING STREAM

List<Employee> sortedList = employee\_list.stream()

.sorted((e1,e2)-> (e1.name).compareTo(e2.name))

.collect(Collectors.toList());

//ALTERNATE TO ABOVE SORTING

// List<Employee> sortedList = employee\_list.stream()

// .sorted(Comparator.comparing(Employee::name))

// .collect(Collectors.toList());

//SORTING WITHOUT LAMDA EXPRESSION

// Collections.sort(employee\_list, new Comparator<Employee>() {

// @Override

// public int compare(Employee first, Employee second)

// {

// return (first.name).compareTo(second.name);

// }

// });

System.out.println("\nEmployee List sorted name wise: \n");

for(Employee e:sortedList)

{

System.out.println("Employee name: "+ e.name+" \nEmployee id: "+e.id+"\n");

}

}

}