**Java Developer Assignment**

Answer1: -

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

public class StringSorting {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a string:");

String input = scanner.nextLine();

String output = sortString(input);

System.out.println("Input: " + input);

System.out.println("Output: " + output);

scanner.close();

}

public static String sortString(String input) {

String[] words = input.split(" ");

Arrays.sort(words, new CustomComparator());

return String.join(" ", words);

}

private static class CustomComparator implements Comparator<String> {

public int compare(String s1, String s2) {

int lengthComparison = Integer.compare(s2.length(), s1.length());

if (lengthComparison != 0) {

return lengthComparison;

}

int occurrencesComparison = Integer.compare(countOccurrences(s2, 'e'), countOccurrences(s1, 'e'));

return occurrencesComparison;

}

}

private static int countOccurrences(String s, char target) {

int count = 0;

for (char c : s.toCharArray()) {

if (c == target) {

count++;

}

}

return count;

}

}

Answer2: -

**import** java.util.\*;

**public** **class** RouteGrouping {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Map<Integer, List<String>> groupedRoutes = *groupRoutesByFare*(*getInputData*());

System.***out***.println("Fare \t\t Route");

**for** (Map.Entry<Integer, List<String>> entry : groupedRoutes.entrySet()) {

System.***out***.println(entry.getKey() + "\t\t" + entry.getValue());

}

}

**private** **static** Map<Integer, List<String>> groupRoutesByFare(String[] inputData) {

Map<Integer, List<String>> fareToRoutesMap = **new** HashMap<>();

**for** (String routeData : inputData) {

String[] parts = routeData.split("\\s+");

**if** (parts.length == 2) {

String route = parts[0];

**int** fare = Integer.*parseInt*(parts[1]);

fareToRoutesMap.computeIfAbsent(fare, k -> **new** ArrayList<>()).add(route);

}

}

**return** fareToRoutesMap;

}

**private** **static** String[] getInputData() {

**return** **new** String[]{

"13 10",

"13-C 15",

"342-R 10",

"146-Q 10",

"27 15",

"29-A 12",

"215-U 12",

"27-E1 15",

"13J 12",

"SBS-D34G 10"

};

}

}

Answer3: -

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

public class AddDaysToDate {

public static void main(String[] args) {

// TODO Auto-generated method stub

if (args.length != 2) {

System.out.println("Usage: java AddDaysToDate <date> <days>");

System.exit(1);

}

String inputDate = args[0];

int daysToAdd = Integer.parseInt(args[1]);

SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");

try {

Date date = sdf.parse(inputDate);

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

calendar.add(Calendar.DAY\_OF\_MONTH, daysToAdd);

String outputDate = sdf.format(calendar.getTime());

System.out.println(outputDate);

} catch (ParseException e) {

System.out.println("Invalid date format. Please use dd-MM-yyyy.");

}

}

}

Answer4: -

**public** **class** Concurrency {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

Test test = **new** Test();

Thread thread1 = **new** Thread(test);

Thread thread2 = **new** Thread(test);

thread1.start();

thread2.start();

}

}

**class** Test **implements** Runnable {

**public** **void** run() {

**for** (**int** i = 1; i <= 5; i++) {

System.***out***.println(Thread.*currentThread*().getId() + " Value " + i);

}

}

}

Answer5: -

create table routes (id INT PRIMARY KEY , name VARCHAR(250), number VARCHAR(250));

INSERT INTO routes (id, name, number)

VALUES (1, '200-D', 'SBV-BHJ'), (2, '300-A', 'ASD-WE');

CREATE TABLE route\_points (id INT PRIMARY KEY , route\_id INT , orders INT , distance INT, FOREIGN KEY (route\_id) REFERENCES routes(id));

INSERT INTO route\_points (id, route\_id, orders, distance)

VALUES

(1, 1, 1, 0),

(2, 1, 2, 100),

(3, 2, 1, 0),

(4, 2, 2, 50),

(5, 2, 3, 100);

SELECT

rp.route\_id,

r.name,

SUM(rp.distance) AS total\_distance

FROM

route\_points rp

JOIN

routes r ON rp.route\_id = r.id

GROUP BY

rp.route\_id, r.name

ORDER BY

total\_distance DESC;

CREATE TABLE station (

id INT PRIMARY KEY,

station\_name VARCHAR(255)

);

Answer6: -

CREATE TABLE times (

id INT PRIMARY KEY,

station\_id INT,

slot INT,

time TIME,

FOREIGN KEY (station\_id) REFERENCES station(id)

);

INSERT INTO station (id, station\_name)

VALUES

(1, 'STA 1'),

(2, 'STB 2');

INSERT INTO times (id, station\_id, slot, time)

VALUES

(1, 1, 1, '6:30'),

(2, 1, 2, '6:45'),

(3, 1, 1, '7:40'),

(4, 1, 2, '7:25'),

(5, 2, 1, '6:40'),

(6, 2, 2, '6:50'),

(7, 2, 1, '7:50'),

(8, 2, 2, '8:15');

SELECT

t.station\_id,

s.station\_name,

t.slot,

t.time

FROM

times t

JOIN

station s ON t.station\_id = s.id

WHERE

t.slot = 1 AND t.time < '07:45'

ORDER BY

t.time;

Answer7: -

CREATE TABLE routes1 (

id INT PRIMARY KEY,

name VARCHAR(255),

number VARCHAR(255)

);

CREATE TABLE route1\_points (

id INT PRIMARY KEY,

route\_id INT,

order\_num INT,

station\_id INT,

stop\_id INT

);

INSERT INTO routes1 (id, name, number)

VALUES

(1, '200-D', 'SBV-BHJ'),

(2, '300-D', 'ASD-WER');

INSERT INTO route1\_points (id, route\_id, order\_num, station\_id, stop\_id)

VALUES

(1, 1, 1, 1, 1),

(2, 1, 2, 1, 2),

(3, 2, 1, 1, 1),

(4, 2, 2, 1, 2),

(5, 2, 3, 1, 3);

SELECT

r.id AS route\_id,

rp1.stop\_id AS source\_stop\_id,

rp2.stop\_id AS dest\_stop\_id

FROM

routes1 r

JOIN

route1\_points rp1 ON r.id = rp1.route\_id AND rp1.order\_num = 1

JOIN

route1\_points rp2 ON r.id = rp2.route\_id AND rp2.order\_num = 2;

Answer8: -

**public** **class** ReversePrintLinkedList {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

LinkedList linkedList = **new** LinkedList();

linkedList.insert(1);

linkedList.insert(2);

linkedList.insert(3);

linkedList.insert(4);

linkedList.insert(5);

System.***out***.print("Linked List elements in reverse order: ");

linkedList.printReverse();

}

}

**class** Node {

**int** data;

Node next;

**public** Node(**int** data) {

**this**.data = data;

**this**.next = **null**;

}

}

**class** LinkedList {

Node head;

**public** **void** insert(**int** data) {

Node newNode = **new** Node(data);

newNode.next = head;

head = newNode;

}

**public** **void** printReverse() {

printReverseRecursive(head);

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

}

**private** **void** printReverseRecursive(Node node) {

**if** (node == **null**) {

**return**;

}

printReverseRecursive(node.next);

System.***out***.print(node.data + " ");

}

}

Answer9: -

**public** **class** SortArray {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[] array = {0, 1, 2, 1, 2, 0, 2, 0, 1};

System.***out***.println("Original array:");

*printArray*(array);

*sortArray*(array);

System.***out***.println("Sorted array:");

*printArray*(array);

}

**public** **static** **void** sortArray(**int**[] nums) {

**int** low = 0;

**int** high = nums.length - 1;

**int** mid = 0;

**while** (mid <= high) {

**switch** (nums[mid]) {

**case** 0:

*swap*(nums, low, mid);

low++;

mid++;

**break**;

**case** 1:

mid++;

**break**;

**case** 2:

*swap*(nums, mid, high);

high--;

**break**;

}

}

}

**private** **static** **void** swap(**int**[] nums, **int** i, **int** j) {

**int** temp = nums[i];

nums[i] = nums[j];

nums[j] = temp;

}

**public** **static** **void** printArray(**int**[] nums) {

**for** (**int** num : nums) {

System.***out***.print(num + " ");

}

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

}

}

Answer10: -

d1.leg = 4

d2.leg = 3

d1.leg = 0

d1 is created and initialized with 4 legs.

d2 is created and initialized with 3 legs.

Swap method takes two Dog objects but doesn't swap their values. The swapping is done locally within the method, and the changes do not affect the original objects.

d1 and d2 within the main method remain unchanged.

Hence output is :

d1.leg = 4

d2.leg = 3

The modify method takes a Dog object and sets its leg property to 0.

Hence output is :

d1.leg = 0