**Java Developer Assignment**

1. Write a program for sorting the string according to the word character length and number of occurrences of a character.

Input - “readable content of a page when looking at its layout”

Output - “readable content looking layout page when its of at a”

public class StringSorter {

public static void main(String[] args) {

String inputStr = "readable content of a page when looking at its layout";

String outputStr = sortString(inputStr);

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

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

}

private static String sortString(String inputStr) {

String[] words = inputStr.split("\\s+"); // Split the input string into an array of words

Arrays.sort(words, Comparator.comparingInt((String word) -> word.length())

.thenComparingInt((String word) -> countOccurrences(word, 'a')));

return String.join(" ", words);

}

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

int count = 0;

for (char ch : word.toCharArray()) {

if (ch == target) {

count++;

}

return count;

}

}

2. Write a program to group by route according to fare without inbuilt libraries.

Input - Route Fare

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

Expected Output - Fare Route

15 ["13-C","27","27-E1"]

10 ["13","342-R","146-Q","SBC-D34G"]

12 ["29-A","215-U","13J"]

public class RouteFareGrouping {

public static void main(String[] args) {

String[] routes = {"13", "13-C", "342-R", "146-Q", "27", "29-A", "215-U", "27-E1", "13J", "SBS-D34G"};

int[] fares = {10, 15, 10, 10, 15, 12, 12, 15, 12, 10};

Map<Integer, List<String>> fareRouteMap = groupRoutesByFare(routes, fares);

System.out.println("Fare\t\tRoute");

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

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

}

}

private static Map<Integer, List<String>> groupRoutesByFare(String[] routes, int[] fares) {

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

for (int i = 0; i < routes.length; i++) {

int fare = fares[i];

String route = routes[i];

fareRouteMap.putIfAbsent(fare, new ArrayList<>());

fareRouteMap.get(fare).add(route);

}

return fareRouteMap;

}

}

3. Write a program to add some days in date which have been taken from command line arguments.

Input – date : 20-05-2022 and days : 5 // 30-05-2022 and days : 5

Output – 25-05-2022 // 04-06-2022

public class AddDaysToDate {

public static void main(String[] args) {

if (args.length != 2) {

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

return;

}

String inputDate = args[0];

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

try {

String newDate = addDays(inputDate, daysToAdd);

System.out.println("Input Date: " + inputDate);

System.out.println("Days to Add: " + daysToAdd);

System.out.println("New Date: " + newDate);

} catch (ParseException e) {

System.out.println("Error parsing date: " + e.getMessage());

}

}

private static String addDays(String inputDate, int daysToAdd) throws ParseException {

DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");

Date date = dateFormat.parse(inputDate);

Calendar calendar = Calendar.getInstance();

calendar.setTime(date);

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

return dateFormat.format(calendar.getTime());

}

}

4. Write a program to achieve concurrency

class MyThread extends Thread {

private final String name;

public MyThread(String name) {

this.name = name;

}

@Override

public void run() {

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

System.out.println(name + ": " + i);

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class ConcurrencyExample {

public static void main(String[] args) {

MyThread thread1 = new MyThread("Thread 1");

MyThread thread2 = new MyThread("Thread 2");

thread1.start();

thread2.start();

}

}

5. Create query for result

Table – routes

| id | name | number |
| --- | --- | --- |
| 1 | 200-D | SBV-BHJ |
| 2 | 300-A | ASD-WE |

Table - route\_points

| id | route\_id | order | distance |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 0 |
| 2 | 1 | 2 | 100 |
| 3 | 2 | 1 | 0 |
| 4 | 2 | 2 | 50 |
| 5 | 2 | 3 | 100 |

Expected Result

| route\_id | name | total\_distance |
| --- | --- | --- |
| 2 | 300-A | 150 |
| 1 | 200-D | 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;

6. Make a query for the result by station\_id and slot. Find all stations where slot 1 and time less than 7.45 and order by time.

Table – station

| id | station\_name |
| --- | --- |
| 1 | STA 1 |
| 2 | STB 2 |

Table – times

| id | station\_id | slot | time |
| --- | --- | --- | --- |
| 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 |

Require columns in output

| station\_id | station\_name | slot | time |
| --- | --- | --- | --- |

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 < '7:45'

ORDER BY

t.time;

7. Make a query for get result for same route by stop id for source and destination

Table – routes

| id | name | number |
| --- | --- | --- |
| 1 | 200-D | SBV-BHJ |
| 2 | 300-D | ASD-WER |

Table - route\_points

| id | station\_id | order | stop\_id |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 2 |
| 3 | 2 | 1 | 1 |
| 4 | 2 | 2 | 2 |
| 5 | 2 | 3 | 3 |

Expected Result

| route\_id | source\_stop\_id | dest\_stop\_id |
| --- | --- | --- |
| 1 | 1 | 2 |
| 2 | 1 | 3 |

Optional: If the above output is achieved. Please try the following

Table – station

| id | station\_name |
| --- | --- |
| 1 | STA 1 |
| 2 | STB 2 |
| 3 | STC 3 |

Expected Result

| route\_id | source\_stop\_name | dest\_stop\_name |
| --- | --- | --- |
| 1 | STA 1 | STB 2 |
| 2 | STA 1 | STC 3 |

WITH RouteStops AS (

SELECT

rp1.route\_id,

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

rp2.stop\_id AS dest\_stop\_id

FROM

route\_points rp1

JOIN

route\_points rp2 ON rp1.route\_id = rp2.route\_id

WHERE

rp1.order = 1

AND rp2.order = (SELECT MAX(order) FROM route\_points WHERE route\_id = rp1.route\_id)

)

SELECT

rs.route\_id,

s1.station\_name AS source\_stop\_name,

s2.station\_name AS dest\_stop\_name

FROM

RouteStops rs

JOIN

station s1 ON rs.source\_stop\_id = s1.id

JOIN

station s2 ON rs.dest\_stop\_id = s2.id;

8. Write a program to implement a linked list in Java and print in reverse order.

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);

}

private void printReverseRecursive(Node node) {

if (node == null) {

return;

}

printReverseRecursive(node.next);

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

}

}

public class Main {

public static void main(String[] args) {

LinkedList linkedList = new LinkedList();

linkedList.insert(1);

linkedList.insert(2);

linkedList.insert(3);

linkedList.insert(4);

linkedList.insert(5);

System.out.println("Original Linked List:");

linkedList.printReverse();

}

}

9. Write a program to sort the given array without any inbuilt functions to achieve O(n).

[0,1,2,1,2,0,2,0,1]

Output –

[0,0,0,1,1,1,2,2,2]

public class SortArrayWithoutInbuiltFunctions {

public static void main(String[] args) {

int[] arr = {0, 1, 2, 1, 2, 0, 2, 0, 1};

System.out.println("Original Array:");

printArray(arr);

sortArray(arr);

System.out.println("\nSorted Array:");

printArray(arr);

}

private static void sortArray(int[] arr) {

int low = 0;

int high = arr.length - 1;

int mid = 0;

while (mid <= high) {

switch (arr[mid]) {

case 0:

swap(arr, low, mid);

low++;

mid++;

break;

case 1:

mid++;

break;

case 2:

swap(arr, mid, high);

high--;

break;

}

}

}

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

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

private static void printArray(int[] arr) {

for (int num : arr) {

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

}

System.out.println();

}

}

10. Give the output for the following and explain.

class Dog {

int leg;

}

class Main {

public static void main(String[] args) {

Dog d1 = new Dog();

d1.leg = 4;

Dog d2 = new Dog();

d2.leg = 3;

swap(d1, d2);

System.out.println("d1.leg = "+d1.leg);

System.out.println("d2.leg = "+d2.leg);

modify(d1);

System.out.println("d1.leg = "+d1.leg);

}

static void swap(Dog d1, Dog d2) {

Dog temp = d2;

d2 = d1;

d1 = temp;

}

static void modify(Dog d1) {

d1.leg = 0;

}

}

Output:

d1.leg = 4

d2.leg = 3

d1.leg = 0

The swap method swaps the local variables d1 and d2, but these changes do not affect the objects outside the method. The modify method directly modifies the leg attribute of the object referred to byd1. As a result, the leg attribute of the d1 object becomes 0. Finally, when printing the leg of d1 in the main method after the modify call, it reflects the modification made in the modify method, resulting in d1.leg = 0.