1-Given a list of custom objects , sort the list based on various parameter of the

object. Suppose you have list of employee object then sort the list based on

name , salary etc.

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

// Employee class

class Employee {

private String name;

private double salary;

private int age;

// Constructor

public Employee(String name, double salary, int age) {

this.name = name;

this.salary = salary;

this.age = age;

}

// Getters

public String getName() {

return name;

}

public double getSalary() {

return salary;

}

public int getAge() {

return age;

}

@Override

public String toString() {

return "Employee{name='" + name + "', salary=" + salary + ", age=" + age + '}';

}

}

public class EmployeeSortExample {

public static void main(String[] args) {

List<Employee> employees = new ArrayList<>();

employees.add(new Employee("John", 50000, 28));

employees.add(new Employee("Alice", 75000, 34));

employees.add(new Employee("Bob", 45000, 25));

employees.add(new Employee("Diana", 60000, 29));

// Sort by name

employees.sort(Comparator.comparing(Employee::getName));

System.out.println("Sorted by Name:");

employees.forEach(System.out::println);

// Sort by salary

employees.sort(Comparator.comparingDouble(Employee::getSalary));

System.out.println("\nSorted by Salary:");

employees.forEach(System.out::println);

// Sort by age (descending)

employees.sort(Comparator.comparingInt(Employee::getAge).reversed());

System.out.println("\nSorted by Age (Descending):");

employees.forEach(System.out::println);

// Sort by multiple criteria (e.g., name, then salary)

employees.sort(Comparator.comparing(Employee::getName)

.thenComparing(Employee::getSalary));

System.out.println("\nSorted by Name and then Salary:");

employees.forEach(System.out::println);

}

}

2-Create a Deadlock situation programmatically.

class DeadLock {

static final String resource1= "Printer";

static final String resource2= "Scanner";

public static void main(String[] args) {

ThreadDemo1 Thread1 = new ThreadDemo1();

ThreadDemo2 Thread2 = new ThreadDemo2();

Thread1.start();

Thread2.start();

}

private static class ThreadDemo1 extends Thread {

public void run() {

synchronized (resource1) {

System.out.println("Desktop "+ ": locked" + resource1);

try {

Thread.sleep(1000);

} catch (Exception e) {

}

}

System.out.println("Desktop " + ": waiting for" + resource2+"......");

synchronized (resource2) {

System.out.println("Desktop "+ ": locked" + resource2);

}

}

}

private static class ThreadDemo2 extends Thread {

public void run() {

synchronized (resource2) {

System.out.println("Laptop "+": locked" + resource2);

try {

Thread.sleep(100);

} catch (Exception e) {

}

}

System.out.println("Laptop " + ": waiting for" + resource1+"......");

synchronized (resource1) {

System.out.println("Laptop "+ ": locked" + resource1);

}

}

}

}

3-Check for Balanced Brackets in an expression.

import java.util.Stack;

public class BalancedBrackets {

**public** **static** **boolean** isBalanced(String exp) {

Stack<Character> stack = **new** Stack<>();

**for** (**char** ch : exp.toCharArray()) {

**if** (ch == '(' || ch == '[' || ch == '{') {

stack.push(ch);

}

**else** **if** (ch == ')' || ch == ']' || ch == '}') {

**if** (stack.isEmpty() || !*isMatchingPair*(stack.pop(), ch)) {

**return** **false**;

}

}

}

**return** stack.isEmpty();

}

**private** **static** **boolean** isMatchingPair(**char** open, **char** close) {

**return** (open == '(' && close == ')') ||

(open == '[' && close == ']') ||

(open == '{' && close == '}');

}

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

String exp1 = "[(0]}{[00]0}";

String exp2 = "[(])";

System.***out***.println("Expression: " + exp1);

System.***out***.println("Output: " + (*isBalanced*(exp1) ? "Balanced" : "Not Balanced"));

System.***out***.println("Expression: " + exp2);

System.***out***.println("Output: " + (*isBalanced*(exp2) ? "Balanced" : "Not Balanced"));

}

}

4-Implement the linked list in java.

**import** java.io.\*;

**public** **class** LinkedList {

Node head;

**static** **class** Node {

**int** data;

Node next;

Node(**int** d)

{

data = d;

next = **null**;

}

}

**public** **static** LinkedList insert(LinkedList list, **int** data)

{

Node new\_node = **new** Node(data);

**if** (list.head == **null**) {

list.head = new\_node;

}

**else** {

Node last = list.head;

**while** (last.next != **null**) {

last = last.next;

}

last.next = new\_node;

}

**return** list;

}

**public** **static** **void** printList(LinkedList list)

{

Node currNode = list.head;

System.***out***.print("LinkedList: ");

**while** (currNode != **null**) {

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

currNode = currNode.next;

}

}

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

{

LinkedList list = **new** LinkedList();

list = *insert*(list, 1);

list = *insert*(list, 2);

list = *insert*(list, 3);

list = *insert*(list, 4);

list = *insert*(list, 5);

list = *insert*(list, 6);

list = *insert*(list, 7);

list = *insert*(list, 8);

*printList*(list);

}

}

5- How do you find the second largest number in an array in Java

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.List;

**import** java.util.Optional;

**public** **class** SecondLargest {

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

List<Integer> numbers = Arrays.*asList*(10, 20, 35, 50, 50, 75, 65);

Optional<Integer> secondLargest = numbers.stream()

.distinct()

.sorted(Comparator.*reverseOrder*())

.skip(1)

.findFirst();

**if** (secondLargest.isPresent()) {

System.***out***.println("The second largest number is: " + secondLargest.get());

} **else** {

System.***out***.println("The list does not have enough unique numbers.");

}

}

}

6- Write a Java program that will print the number of occurrences of each

character in a string.

import java.util.Map;

import java.util.function.Function;

import java.util.stream.Collectors;

public class CharacterFrequency {

public static void main(String[] args) {

String input= "PRAVEEN";

Map<Character, Long> characterFrequency = input.chars()

.filter(c -> c !=' ')

.mapToObj(c -> (char) c)

.collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));

characterFrequency.forEach((character, frequency)->

System.out.println(" ' " + character + " ' ->" + frequency));

}

}

7- Write java program to shift the element to the right by one index.

import java.util.Arrays;

public class ShiftRight {

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

**int**[] arr = {1, 2, 3, 4, 5};

// Shift elements to the right by one index

**int** lastElement = arr[arr.length - 1];

**for** (**int** i = arr.length - 1; i > 0; i--) {

arr[i] = arr[i - 1];

}

arr[0] = lastElement;

System.***out***.println(Arrays.*toString*(arr));

}

}

React Coding Questions:

1. Write a JavaScript function to return Fibonacci numbers till n.

function fibonacci(n) {

const fibNumbers = [0, 1];

for (let i = 2; i <= n; i++) {

fibNumbers[i] = fibNumbers[i - 1] + fibNumbers[i - 2];

}

return fibNumbers.slice(0, n + 1);

}

const n = 10;

const result = fibonacci(n);

console.log(result);

2- In the array, [1,1,2,2,2,3,3,4,5,5,5]; write a Javascript function to show the

elements with no of repetitions.

function countOccurrences(arr) {

const counts = {};

arr.forEach(num => {

counts[num] = (counts[num] || 0) + 1;

});

for (const [num, count] of Object.entries(counts)) {

console.log(`${num} occurs ${count} times`);

}

}

const arr = [1, 1, 2, 2, 2, 3, 3, 4, 5, 5, 5];

countOccurrences(arr);

3-Write a javascript function to check if a string is palindrome or not?

function isPalindrome(str) {

const cleanedStr = str.toLowerCase().replace(/[^a-z0-9]/g,

'');

return cleanedStr === cleanedStr.split('').reverse().join('');

}

const str1 = "racecar";

const str2 = "A man, a plan, a canal: Panama";

const str3 = "hello world";

console.log(isPalindrome(str1));

console.log(isPalindrome(str2));

console.log(isPalindrome(str3));