37. public class Main {

public static double myPow(double x, int n) {

if (n == 0) return 1;

if (n < 0) {

x = 1 / x;

n = -n;

}

double result = 1;

while (n > 0) {

if (n % 2 == 1) {

result \*= x;

}

x \*= x;

n /= 2;

}

return result;

}

public static void main(String[] args) {

double x = 2.00000;

int n = 10;

System.out.println(myPow(x, n)); // Output: 1024.00000

}

}

38. public class Main {

public static int maxSubArray(int[] nums) {

if (nums == null || nums.length == 0) {

return 0;

}

int maxSum = nums[0];

int currentSum = nums[0];

for (int i = 1; i < nums.length; i++) {

currentSum = Math.max(nums[i], currentSum + nums[i]);

maxSum = Math.max(maxSum, currentSum);

}

return maxSum;

}

public static void main(String[] args) {

int[] nums = {-2, 1, -3, 4, -1, 2, 1, -5, 4};

System.out.println(maxSubArray(nums)); // Output: 6

}

}

39. public class Singleton {

private static Singleton instance = null;

private Singleton() {

// Private constructor to prevent instantiation from outside

}

public static Singleton getInstance() throws IllegalAccessException {

if (instance == null) {

instance = new Singleton();

return instance;

} else {

throw new IllegalAccessException("Only one instance of Singleton is allowed.");

}

}

public static void main(String[] args) {

try {

Singleton obj1 = Singleton.getInstance();

System.out.println("Object 1 created successfully.");

Singleton obj2 = Singleton.getInstance(); // This should throw an exception

System.out.println("Object 2 created successfully.");

} catch (IllegalAccessException e) {

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

}

}

}

40. import java.util.TreeSet;

class ExamRoom {

private int n;

private TreeSet<Integer> students;

public ExamRoom(int n) {

this.n = n;

this.students = new TreeSet<>();

}

public int seat() {

if (students.isEmpty()) {

students.add(0);

return 0;

}

int maxDistance = students.first();

Integer prev = null;

int seat = 0;

for (int student : students) {

if (prev != null) {

int distance = (student - prev) / 2;

if (distance > maxDistance) {

maxDistance = distance;

seat = prev + distance;

}

}

prev = student;

}

if (n - 1 - students.last() > maxDistance) {

seat = n - 1;

}

students.add(seat);

return seat;

}

public void leave(int p) {

students.remove(p);

}

public static void main(String[] args) {

ExamRoom examRoom = new ExamRoom(10);

System.out.println(examRoom.seat()); // Output: 0

System.out.println(examRoom.seat()); // Output: 9

System.out.println(examRoom.seat()); // Output: 4

System.out.println(examRoom.seat()); // Output: 2

examRoom.leave(4);

System.out.println(examRoom.seat()); // Output: 5

}

}

41. import java.util.HashMap;

import java.util.Map;

public class Main {

public static int numTilePossibilities(String tiles) {

Map<Character, Integer> freqMap = new HashMap<>();

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

freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);

}

return dfs(freqMap);

}

private static int dfs(Map<Character, Integer> freqMap) {

int sum = 0;

for (char c : freqMap.keySet()) {

int count = freqMap.get(c);

if (count == 0) continue; // Skip if count is already used

sum++; // Count the sequence consisting of this character only

freqMap.put(c, count - 1); // Mark this character as used

sum += dfs(freqMap); // Count sequences with remaining characters

freqMap.put(c, count); // Backtrack: restore count of this character

}

return sum;

}

public static void main(String[] args) {

String tiles = "AAB";

System.out.println(numTilePossibilities(tiles)); // Output: 8

}

}

42. import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileCopy {

public static void main(String[] args) {

String sourceFileName = "source.txt"; // Name of the source file

String destinationFileName = "destination.txt"; // Name of the destination file

try (BufferedReader reader = new BufferedReader(new FileReader(sourceFileName));

BufferedWriter writer = new BufferedWriter(new FileWriter(destinationFileName))) {

String line;

while ((line = reader.readLine()) != null) {

writer.write(line);

writer.newLine(); // Add a newline after each line copied

}

System.out.println("File copied successfully.");

} catch (IOException e) {

System.err.println("Error reading or writing file: " + e.getMessage());

}

}

}

43. import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileCopy {

public static void main(String[] args) {

String sourceFileName = "source.txt"; // Name of the source file

String destinationFileName = "destination.txt"; // Name of the destination file

try (FileReader reader = new FileReader(sourceFileName);

FileWriter writer = new FileWriter(destinationFileName)) {

int character;

while ((character = reader.read()) != -1) {

writer.write(character);

}

System.out.println("File copied successfully.");

} catch (IOException e) {

System.err.println("Error reading or writing file: " + e.getMessage());

}

}

}

44. import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ReadFile {

public static void main(String[] args) {

String fileName = "example.txt"; // Name of the file to read

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line); // Print each line read from the file

}

} catch (IOException e) {

System.err.println("Error reading file: " + e.getMessage());

}

}

}

45. import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class FileStatsCounter {

public static void main(String[] args) {

String fileName = "input.txt"; // Change this to your file name

int charCount = 0;

int wordCount = 0;

int lineCount = 0;

try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = br.readLine()) != null) {

lineCount++;

charCount += line.length();

String[] words = line.trim().split("\\s+");

wordCount += words.length;

}

} catch (IOException e) {

e.printStackTrace();

}

System.out.println("Character count: " + charCount);

System.out.println("Word count: " + wordCount);

System.out.println("Line count: " + lineCount);

}

}

46. import java.util.Scanner;

public class MatrixDisplay {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the matrix (N): ");

int n = scanner.nextInt();

int[][] matrix = new int[n][n];

// Input the matrix

System.out.println("Enter the elements of the matrix:");

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

for (int j = 0; j < n; j++) {

matrix[i][j] = scanner.nextInt();

}

}

// Display matrix row-wise

System.out.println("Matrix displayed row-wise:");

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

for (int j = 0; j < n; j++) {

System.out.print(matrix[i][j] + " ");

}

System.out.println();

}

// Display matrix column-wise

System.out.println("Matrix displayed column-wise:");

for (int j = 0; j < n; j++) {

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

System.out.print(matrix[i][j] + " ");

}

System.out.println();

}

scanner.close();

}

}

47. // Interface definition

interface IterF1 {

void add(int a, int b);

void sub(int a, int b);

}

// Class implementing the interface

class Calculator implements IterF1 {

@Override

public void add(int a, int b) {

int result = a + b;

System.out.println("Addition result: " + result);

}

@Override

public void sub(int a, int b) {

int result = a - b;

System.out.println("Subtraction result: " + result);

}

}

// Main class

public class Main {

public static void main(String[] args) {

Calculator calculator = new Calculator();

// Overloaded add method

calculator.add(5, 3);

// Overloaded sub method

calculator.sub(10, 4);

}

}

48. import java.util.Scanner;

public class EPFInterestCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter the EPF balance: ");

double balance = scanner.nextDouble();

System.out.print("Enter the number of years: ");

int years = scanner.nextInt();

double interestRate = calculateInterest(balance, years);

System.out.println("Interest rate: " + interestRate + "%");

} catch (Exception e) {

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

} finally {

scanner.close();

}

}

public static double calculateInterest(double balance, int years) {

if (balance <= 0 || years <= 0) {

throw new IllegalArgumentException("Balance and years must be positive values.");

}

double interestRate;

if (years < 5) {

interestRate = 5.0; // 5% interest for less than 5 years

} else {

interestRate = 8.0; // 8% interest for 5 years or more

}

return interestRate;

}

}

49. class MyThread extends Thread {

MyThread() {

super(); // Call base class constructor

start(); // Start the thread

}

public void run() {

// This code will be executed when the thread starts

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

System.out.println("Child Thread: " + i);

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

public class Main {

public static void main(String[] args) {

MyThread myThread = new MyThread(); // Create an instance of MyThread

// Main thread continues its execution concurrently with the child thread

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

System.out.println("Main Thread: " + i);

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

50. class AgeNotWithinRangeException extends Exception {

public AgeNotWithinRangeException(String message) {

super(message);

}

}

class NameNotValidException extends Exception {

public NameNotValidException(String message) {

super(message);

}

}

class Student {

private int rollNo;

private String name;

private int age;

private String course;

public Student(int rollNo, String name, int age, String course) throws AgeNotWithinRangeException, NameNotValidException {

if (age < 15 || age > 21) {

throw new AgeNotWithinRangeException("Age should be between 15 and 21");

}

if (!name.matches("[a-zA-Z\\s]+")) {

throw new NameNotValidException("Name should not contain numbers or special symbols");

}

this.rollNo = rollNo;

this.name = name;

this.age = age;

this.course = course;

}

public void display() {

System.out.println("Roll No: " + rollNo);

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

System.out.println("Age: " + age);

System.out.println("Course: " + course);

}

}

public class Main {

public static void main(String[] args) {

try {

Student student1 = new Student(101, "John Doe", 20, "Computer Science");

student1.display();

System.out.println();

// This will throw AgeNotWithinRangeException

Student student2 = new Student(102, "Alice", 25, "Mathematics");

} catch (AgeNotWithinRangeException | NameNotValidException e) {

System.out.println("Exception: " + e.getMessage());

}

}

}