1. Write a program to count all the prime and composite numbers entered by the user.

Sample Input:

Enter the numbers

4

54

29

71

7

59

98

23

Sample Output:

Composite number:3

Prime number:5

Test cases:

1. 33, 41, 52, 61,73,90
2. TEN, FIFTY, SIXTY-ONE, SEVENTY-SEVEN, NINE
3. 45, 87, 09, 5.0 ,2.3, 0.4
4. -54, -76, -97, -23, -33, -98
5. 45, 73, 00, 50, 67, 44

**ANSWER:**

import java.util.Scanner;

public class PrimeCompositeCounter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the numbers (separate with spaces): ");

String[] inputNumbers = scanner.nextLine().split(" ");

int primeCount = 0;

int compositeCount = 0;

for (String inputNumber : inputNumbers) {

try {

int number = Integer.parseInt(inputNumber);

if (isPrime(number)) {

primeCount++;

} else {

compositeCount++;

}

} catch (NumberFormatException e) {

}

}

System.out.println("Composite number: " + compositeCount);

System.out.println("Prime number: " + primeCount);

scanner.close();

}

private static boolean isPrime(int n) {

if (n <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(n); i++) {

if (n % i == 0) {

return false;

}

}

return true;

}

}

1. **Find the Mth maximum number and Nth minimum number in an array and then find the sum of it and difference of it.**

**Sample Input:**

**Array of elements = {14, 16, 87, 36, 25, 89, 34}**

**M = 1**

**N = 3**

**Sample Output:**

**1stMaximum Number = 89**

**3rdMinimum Number = 25**

**Sum = 114**

**Difference = 64**

**Test cases:**

1. **{16, 16, 16 16, 16}, M = 0, N = 1**
2. **{0, 0, 0, 0}, M = 1, N = 2**
3. **{-12, -78, -35, -42, -85}, M = 3 , N = 3**
4. **{15, 19, 34, 56, 12}, M = 6 , N = 3**

**{85, 45, 65, 75, 95}, M = 5 , N = 7**

**ANSWER:**

import java.util.Arrays;

public class MaxMinSumDifference {

public static void main(String[] args) {

int[] array1 = {14, 16, 87, 36, 25, 89, 34};

int m1 = 1;

int n1 = 3;

findMaxMinSumDifference(array1, m1, n1);

int[] array2 = {16, 16, 16, 16, 16};

int m2 = 0;

int n2 = 1;

findMaxMinSumDifference(array2, m2, n2);

int[] array3 = {0, 0, 0, 0};

int m3 = 1;

int n3 = 2;

findMaxMinSumDifference(array3, m3, n3);

int[] array4 = {-12, -78, -35, -42, -85};

int m4 = 3;

int n4 = 3;

findMaxMinSumDifference(array4, m4, n4);

int[] array5 = {15, 19, 34, 56, 12};

int m5 = 6;

int n5 = 3;

findMaxMinSumDifference(array5, m5, n5);

int[] array6 = {85, 45, 65, 75, 95};

int m6 = 5;

int n6 = 7;

findMaxMinSumDifference(array6, m6, n6);

}

public static void findMaxMinSumDifference(int[] array, int m, int n) {

if (m <= 0 || n <= 0 || m > array.length || n > array.length) {

System.out.println("Invalid values of M and N.");

return;

}

Arrays.sort(array);

int mthMax = array[array.length - m];

int nthMin = array[n - 1];

int sum = mthMax + nthMin;

int difference = Math.abs(mthMax - nthMin);

System.out.println("Sorted Array: " + Arrays.toString(array));

System.out.println(m + "th Maximum Number = " + mthMax);

System.out.println(n + "th Minimum Number = " + nthMin);

System.out.println("Sum = " + sum);

System.out.println("Difference = " + difference);

System.out.println();

}

}

1. **Write a program to print the total amount available in the ATM machine with the conditions applied.**

**Total denominations are 2000, 500, 200, 100, get the denomination priority from the user and the total number of notes from the user to display the total available balance to the user**

**Sample Input:**

**Enter the 1st Denomination: 500**

**Enter the 1st Denomination number of notes: 4**

**Enter the 2nd Denomination: 100**

**Enter the 2nd Denomination number of notes: 20**

**Enter the 3rd Denomination: 200**

**Enter the 3rd Denomination number of notes: 32**

**Enter the 4th Denomination: 2000**

**Enter the 4th Denomination number of notes: 1**

**Sample Output:**

**Total Available Balance in ATM: 12400**

**Test Cases:**

**3 Hidden Test cases (Think Accordingly based on Denominations)**

**ANSWER:**

import java.util.Scanner;

public class ATMBalance {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int totalDenominations = 4;

int[] denominations = new int[totalDenominations];

int[] notes = new int[totalDenominations];

System.out.println("Enter the denominations and number of notes:");

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

System.out.print("Enter the " + (i + 1) + " Denomination: ");

denominations[i] = scanner.nextInt();

System.out.print("Enter the " + (i + 1) + " Denomination number of notes: ");

notes[i] = scanner.nextInt();

}

int totalBalance = calculateTotalBalance(denominations, notes);

System.out.println("Total Available Balance in ATM: " + totalBalance);

scanner.close();

}

public static int calculateTotalBalance(int[] denominations, int[] notes) {

int totalBalance = 0;

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

totalBalance += denominations[i] \* notes[i];

}

return totalBalance;

}

}

1. **Write a program using choice to check**

**Case 1: Given string is palindrome or not**

**Case 2: Given number is palindrome or not**

**Sample Input:**

**Case = 1**

**String = MADAM**

**Sample Output:**

**Palindrome**

**Test cases:**

1. **MONEY**
2. **5678765**
3. **MALAY12321ALAM**
4. **MALAYALAM**

**1234.4321**

**ANSWER:**

import java.util.Scanner;

public class PalindromeChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Choose an option:");

System.out.println("1. Check if a string is a palindrome");

System.out.println("2. Check if a number is a palindrome");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

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

String inputString = scanner.nextLine();

if (isPalindrome(inputString)) {

System.out.println("Palindrome");

} else {

System.out.println("Not a Palindrome");

}

break;

case 2:

System.out.print("Enter a number: ");

long inputNumber = scanner.nextLong();

if (isPalindrome(inputNumber)) {

System.out.println("Palindrome");

} else {

System.out.println("Not a Palindrome");

}

break;

default:

System.out.println("Invalid choice.");

}

scanner.close();

}

public static boolean isPalindrome(String str) {

str = str.toLowerCase();

str = str.replaceAll("[^a-z0-9]", "");

characters

StringBuilder reversed = new StringBuilder(str).reverse();

return str.equals(reversed.toString());

}

public static boolean isPalindrome(long num) {

String strNum = String.valueOf(num);

return isPalindrome(strNum);

}

}

1. **Write a program to convert Decimal number equivalent to Binary number and octal numbers?**

**Sample Input:**

**Decimal Number: 15**

**Sample Output:**

**Binary Number = 1111**

**Octal = 17**

**Test cases:**

1. **111**
2. **15.2**
3. **0**
4. **B12**

**5.1A.2**

**ANSWER:**

import java.util.Scanner;

public class DecimalToBinaryOctalConverter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Decimal Number: ");

if (scanner.hasNextInt()) {

int decimalNumber = scanner.nextInt();

String binaryNumber = convertToBinary(decimalNumber);

String octalNumber = convertToOctal(decimalNumber);

System.out.println("Binary Number = " + binaryNumber);

System.out.println("Octal = " + octalNumber);

} else {

System.out.println("Invalid input. Please enter a valid decimal number.");

}

scanner.close();

}

public static String convertToBinary(int decimalNumber) {

return Integer.toBinaryString(decimalNumber);

}

public static String convertToOctal(int decimalNumber) {

return Integer.toOctalString(decimalNumber);

}

}

1. **n an organization they decide to give bonus to all the employees on New Year. A 5% bonus on salary is given to the grade A workers and 10% bonus on salary to the grade B workers. Write a program to enter the salary and grade of the employee. If the salary of the employee is less than $10,000 then the employee gets an extra 2% bonus on salary Calculate the bonus that has to be given to the employee and print the salary that the employee will get.**

**Sample Input & Output:**

**Enter the grade of the employee: B**

**Enter the employee salary: 50000**

**Salary=50000**

**Bonus=5000.0**

**Total to be paid:55000.0**

**Test cases:**

1. **Enter the grade of the employee: A**

**Enter the employee salary: 8000**

1. **Enter the grade of the employee: C**

**Enter the employee salary: 60000**

1. **Enter the grade of the employee: B**

**Enter the employee salary: 0**

1. **Enter the grade of the employee: 38000**

**Enter the employee salary: A**

1. **Enter the grade of the employee: B**

**Enter the employee salary: -8000**

**ANSWER:**

import java.util.Scanner;

public class EmployeeBonusCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the grade of the employee: ");

char grade = scanner.next().charAt(0);

scanner.nextLine();

System.out.print("Enter the employee salary: ");

double salary = scanner.nextDouble();

if (salary < 0) {

System.out.println("Invalid salary. Please enter a valid positive salary.");

} else {

double bonus = calculateBonus(salary, grade);

double totalSalary = salary + bonus;

System.out.println("Salary=" + salary);

System.out.println("Bonus=" + bonus);

System.out.println("Total to be paid: " + totalSalary);

}

scanner.close();

}

public static double calculateBonus(double salary, char grade) {

double bonusPercentage;

if (salary < 10000) {

bonusPercentage = 2;

} else {

switch (grade) {

case 'A':

bonusPercentage = 5;

break;

case 'B':

bonusPercentage = 10;

break;

default:

bonusPercentage = 0;

System.out.println("Invalid grade. No bonus applied.");

}

}

return (salary \* bonusPercentage) / 100;

}

}

1. **Write a program to print the first n perfect numbers. (Hint Perfect number means a positive integer that is equal to the sum of its proper divisors)**

**Sample Input:**

**N = 3**

**Sample Output:**

**First 3 perfect numbers are: 6 , 28 , 496**

**Test Cases:**

1. **N = 0**
2. **N = 5**
3. **N = -2**
4. **N = -5**
5. **N = 0.2**

**ANSWER:**

import java.util.Scanner;

public class PerfectNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

if (n <= 0) {

System.out.println("Invalid input. Please enter a positive integer for N.");

} else {

System.out.print("First " + n + " perfect numbers are: ");

printPerfectNumbers(n);

}

scanner.close();

}

public static void printPerfectNumbers(int n) {

int count = 0;

int number = 1;

while (count < n) {

if (isPerfectNumber(number)) {

System.out.print(number + " , ");

count++;

}

number++;

}

}

public static boolean isPerfectNumber(int num) {

if (num <= 1) {

return false;

}

int sum = 1; // 1 is always a divisor

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

sum += i;

if (i != num / i) {

sum += num / i;

}

}

}

return sum == num;

}

}

1. **Write a program to enter the marks of a student in four subjects. Then calculate the total and aggregate, display the grade obtained by the student. If the student scores an aggregate greater than 75%, then the grade is Distinction. If aggregate is 60>= and <75, then the grade is First Division. If aggregate is 50 >= and <60, then the grade is Second Division. If aggregate is 40>= and <50, then the grade is Third Division. Else the grade is Fail.**

**Sample Input & Output:**

**Enter the marks in python: 90**

**Enter the marks in c programming: 91**

**Enter the marks in Mathematics: 92**

**Enter the marks in Physics: 93**

**Total= 366**

**Aggregate = 91.5**

**DISTINCTION**

**Test cases:**

1. **18, 76,93,65**
2. **73,78,79,75**
3. **98,106,120,95**
4. **96,73, -85,95**
5. **78,59.8,76,79**

**ANSWER:**

import java.util.Scanner;

public class StudentGradeCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the marks in Python: ");

double pythonMarks = scanner.nextDouble();

System.out.print("Enter the marks in C Programming: ");

double cProgrammingMarks = scanner.nextDouble();

System.out.print("Enter the marks in Mathematics: ");

double mathematicsMarks = scanner.nextDouble();

System.out.print("Enter the marks in Physics: ");

double physicsMarks = scanner.nextDouble();

double total = calculateTotal(pythonMarks, cProgrammingMarks, mathematicsMarks, physicsMarks);

double aggregate = calculateAggregate(total);

String grade = calculateGrade(aggregate);

System.out.println("Total= " + total);

System.out.println("Aggregate = " + aggregate);

System.out.println(grade);

scanner.close();

}

public static double calculateTotal(double pythonMarks, double cProgrammingMarks, double mathematicsMarks, double physicsMarks) {

return pythonMarks + cProgrammingMarks + mathematicsMarks + physicsMarks;

}

public static double calculateAggregate(double total) {

return total / 4;

}

public static String calculateGrade(double aggregate) {

if (aggregate > 75) {

return "DISTINCTION";

} else if (aggregate >= 60 && aggregate < 75) {

return "First Division";

} else if (aggregate >= 50 && aggregate < 60) {

return "Second Division";

} else if (aggregate >= 40 && aggregate < 50) {

return "Third Division";

} else {

return "Fail";}

}

}

1. **Write a program to read the numbers until -1 is encountered. Find the average of positive numbers and negative numbers entered by user.**

**Sample Input:**

**Enter -1 to exit…**

**Enter the number: 7**

**Enter the number: -2**

**Enter the number: 9**

**Enter the number: -8**

**Enter the number: -6**

**Enter the number: -4**

**Enter the number: 10**

**Enter the number: -1**

**Sample Output:**

**The average of negative numbers is: -5.0**

**The average of positive numbers is : 8.66666667**

**Test cases:**

1. **-1,43, -87, -29, 1, -9**
2. **73, 7-6,2,10,28,-1**
3. **-5, -9, -46,2,5,0**
4. **9, 11, -5, 6, 0,-1**
5. **-1,-1,-1,-1,-1**

**ANSWER:**

import java.util.Scanner;

public class AverageCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int positiveSum = 0;

int positiveCount = 0;

int negativeSum = 0;

int negativeCount = 0;

System.out.println("Enter -1 to exit...");

while (true) {

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

int number = scanner.nextInt();

if (number == -1) {

break;

}

if (number >= 0) {

positiveSum += number;

positiveCount++;

} else {

negativeSum += number;

negativeCount++;

}

}

if (positiveCount > 0) {

double positiveAverage = (double) positiveSum / positiveCount;

System.out.println("The average of positive numbers is: " + positiveAverage);

} else {

System.out.println("No positive numbers entered.");

}

if (negativeCount > 0) {

double negativeAverage = (double) negativeSum / negativeCount;

System.out.println("The average of negative numbers is: " + negativeAverage);

} else {

System.out.println("No negative numbers entered.");

}

scanner.close();

}

}

1. **Write a program to read a character until a \* is encountered. Also count the number of uppercase, lowercase, and numbers entered by the users.**

**Sample Input:**

**Enter \* to exit…**

**Enter any character: W**

**Enter any character: d**

**Enter any character: A**

**Enter any character: G**

**Enter any character: g**

**Enter any character: H**

**Enter any character: \***

**Sample Output:**

**Total count of lower case:2**

**Total count of upper case:4**

**Total count of numbers =0**

**Test cases:**

1. **1,7,6,9,5**
2. **S, Q, l, K,7, j, M**
3. **M, j, L, &, @, G**
4. **D, K, I, 6, L, \***
5. **\*, K, A, e, 1, 8, %, \***

**ANSWER:**

import java.util.Scanner;

public class CharacterCounter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int lowercaseCount = 0;

int uppercaseCount = 0;

int numberCount = 0;

System.out.println("Enter \* to exit...");

while (true) {

System.out.print("Enter any character: ");

char character = scanner.next().charAt(0);

if (character == '\*') {

break;

}

if (Character.isLowerCase(character)) {

lowercaseCount++;

} else if (Character.isUpperCase(character)) {

uppercaseCount++;

} else if (Character.isDigit(character)) {

numberCount++;

}

}

System.out.println("Total count of lowercase: " + lowercaseCount);

System.out.println("Total count of uppercase: " + uppercaseCount);

System.out.println("Total count of numbers: " + numberCount);

scanner.close();

}

}

1. **Write a program to calculate the factorial of number using recursive function.**

**Sample Input & Output:**

**Enter the value of n: 6**

**Sample Input & Output:**

**The factorial of 6 is: 720**

**Test cases:**

1. **N = 0**
2. **N = -5**
3. **N = 1**
4. **N = M**
5. **N = %**

**ANSWER:**

import java.util.Scanner;

public class FactorialCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the value of n: ");

int n = scanner.nextInt();

if (n < 0) {

System.out.println("Invalid input. Please enter a non-negative integer.");

} else {

long factorial = calculateFactorial(n);

System.out.println("The factorial of " + n + " is: " + factorial);

}

scanner.close();

}

public static long calculateFactorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* calculateFactorial(n - 1);

}

}

}

1. **Write a Program to Find the Nth Largest Number in a array.**

**Sample Input:**

**List : {14, 67, 48, 23, 5, 62}**

**N = 4**

**Sample Output:**

**4th Largest number: 23**

**Test cases:**

1. **N = 0**
2. **N = -5**
3. **N = 1**
4. **N = M**

**5.N = %**

**ANSWER:**

import java.util.Arrays;

import java.util.Scanner;

public class NthLargestNumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int size = scanner.nextInt();

if (size <= 0) {

System.out.println("Invalid input. Please enter a positive size.");

} else {

int[] array = new int[size];

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

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

array[i] = scanner.nextInt();

}

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

int n = scanner.nextInt();

if (n < 1 || n > size) {

System.out.println("Invalid input. N should be between 1 and " + size);

} else {

int nthLargest = findNthLargest(array, n);

System.out.println(n + "th Largest number: " + nthLargest);

}

}

scanner.close();

}

public static int findNthLargest(int[] array, int n) {

Arrays.sort(array);

return array[array.length - n];

}

}

1. **Write a program to convert the Binary to Decimal, Octal**

**Sample Input:**

**Given Number: 1101**

**Sample Output:**

**Decimal Number: 13**

**Octal:15**

**Test cases:**

1. **211**
2. **11011**
3. **22122**
4. **111011.011**

**ANSWER:**

import java.util.Scanner;

public class BinaryConverter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Given Number (Binary): ");

String binaryNumber = scanner.next();

if (isValidBinary(binaryNumber)) {

int decimalNumber = convertToDecimal(binaryNumber);

String octalNumber = convertToOctal(binaryNumber);

System.out.println("Decimal Number: " + decimalNumber);

System.out.println("Octal: " + octalNumber);

} else {

System.out.println("Invalid binary number. Please enter a valid binary number.");

}

scanner.close();

}

public static boolean isValidBinary(String binaryNumber) {

return binaryNumber.matches("[01]+");

}

public static int convertToDecimal(String binaryNumber) {

return Integer.parseInt(binaryNumber, 2);

}

public static String convertToOctal(String binaryNumber) {

int decimalNumber = Integer.parseInt(binaryNumber, 2);

return Integer.toOctalString(decimalNumber);

}

}

1. **Write a program to find the number of special characters in the given statement**

**Sample Input:**

**Given statement: Modi Birthday @ September 17, #&$% is the wishes code for him.**

**Sample Output:**

**Number of special Characters: 5**

**ANSWER:**

import java.util.Scanner;

public class SpecialCharacterCounter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Given statement: ");

scanner.nextLine(); // Consume the newline character left by previous input

String statement = scanner.nextLine();

int specialCharacterCount = countSpecialCharacters(statement);

System.out.println("Number of special characters: " + specialCharacterCount);

scanner.close();

}

public static int countSpecialCharacters(String statement) {

int count = 0;

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

char character = statement.charAt(i);

if (!Character.isLetterOrDigit(character) && !Character.isWhitespace(character))

{

count++;

}

}

return count;

}

}

1. **Write a Program to Remove the Duplicate Items from a array.**

**Sample Input:**

**Enter the number of elements in array:7**

**Enter element1:10**

**Enter element2:20**

**Enter element3:20**

**Enter element4:30**

**Enter element5:40**

**Enter element6:40**

**Enter element7:50**

**Sample Output:**

**Non-duplicate items:**

**[10, 20, 30, 40, 50]**

**ANSWER:**

import java.util.Arrays;

import java.util.Scanner;

public class RemoveDuplicatesFromArray {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] array = new int[n];

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

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

System.out.print("Enter element" + (i + 1) + ": ");

array[i] = scanner.nextInt();

}

int[] uniqueArray = removeDuplicates(array);

System.out.println("Non-duplicate items:");

System.out.println(Arrays.toString(uniqueArray));

scanner.close();

}

public static int[] removeDuplicates(int[] array) {

int n = array.length;

int uniqueCount = 0;

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

boolean isDuplicate = false;

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

if (array[i] == array[j]) {

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

uniqueCount++;

}

}

int[] uniqueArray = new int[uniqueCount];

int index = 0;

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

boolean isDuplicate = false;

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

if (array[i] == array[j]) {

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

uniqueArray[index++] = array[i];

}

}

return uniqueArray;

}

}

1. **Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest. Write a Java program for above scenario.**

**Sample Input SBI, 8.4**

**Sample Output**

**Test case**

**1.SBI, 8.3**

**2.ICICI, 7.3**

**3.AXIS, 9.7**

**4.SBI, 8.6**

**5.AXIX, 7.6**

**ANSWER:**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Bank {

private static final Map<String, Double> interestRates = new HashMap<>();

static {

interestRates.put("SBI", 8.4);

interestRates.put("ICICI", 7.3);

interestRates.put("AXIS", 9.7);

}

public static double getInterestRate(String bankName) {

return interestRates.getOrDefault(bankName, 0.0);

}

}

public class BankInterestExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the bank name: ");

String bankName = scanner.nextLine();

double interestRate = Bank.getInterestRate(bankName);

if (interestRate != 0.0) {

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

} else {

System.out.println("Invalid bank name or no interest rate available for the specified bank.");

}

scanner.close();

}

}

1. **Bring out the situation in which member names of a subclass hide members by the same name in the super class. How it can be resolved? Write Suitable code in Java and**

**Implement above scenario with the Parametrized Constructor (accept int type parameter) of the Super Class can be called from Sub Class Using super () and display the input values provided.**

**Sample Input : 100, 200**

**Sample Output : 100, 200**

**Test Cases**

1. **10, 20**
2. **-20, -30**
3. **0, 0**
4. **EIGHT FIVE**
5. **10.57, 12.58**

**ANSWER:**

class Superclass {

int value;

Superclass(int value) {

this.value = value;

}

void display() {

System.out.println("Value in Superclass: " + value);

}

}

class Subclass extends Superclass {

int value;

Subclass(int value, int subclassValue) {

super(value);

this.value = subclassValue;

}

void display() {

super.display();

System.out.println("Value in Subclass: " + value);

}

}

public class Main {

public static void main(String[] args) {

Subclass obj1 = new Subclass(10, 20);

obj1.display();

Subclass obj2 = new Subclass(-20, -30);

obj2.display();

Subclass obj3 = new Subclass(0, 0);

obj3.display();

incompatible types

incompatible types

}

}

1. **Display Multiplication table for 5 and 10 using various stages of life cycle of the thread by generating a suitable code in Java.**

**Sample Input 5, 10**

**5 X 1 = 5**

**5 X 2 =10**

**10 X 1 =10**

**10 X 2 = 20**

**Test Cases:**

1. **10, 20**
2. **-10, -30**
3. **0, 0**
4. **SIX, SIX**
5. **9.8, 9.6**

**ANSWER:**

class MultiplicationTableThread extends Thread {

private final int number;

public MultiplicationTableThread(int number) {

this.number = number;}

public void run() {

for (int i = 1; i <= 10; i++) {

System.out.println(number + " X " + i + " = " + (number \* i));}

}

}

public class ThreadExample {

public static void main(String[] args) {

MultiplicationTableThread thread1 = new MultiplicationTableThread(5);

thread1.start();

MultiplicationTableThread thread2 = new MultiplicationTableThread(-10);

thread2.start();

MultiplicationTableThread thread3 = new MultiplicationTableThread(0);

thread3.start();

}

}

1. **Using the concepts of thread with implementing Runnable interface in Java to generate Fibonacci series.**

**Sample Input : 5**

**Sample Output : 0 1 1 2 3 …..**

**Test Cases**

1. **7**
2. **-10**
3. **0**
4. **EIGHT FIVE**
5. **12.65**

**ANSWER:**

class FibonacciGenerator implements Runnable {

private final int n;

public FibonacciGenerator(int n) {

this.n = n;

}

public void run() {

int first = 0, second = 1;

System.out.println("Fibonacci series:");

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

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

int next = first + second;

first = second;

second = next;

}

}

}

public class ThreadFibonacciExample {

public static void main(String[] args) {

Thread thread1 = new Thread(new FibonacciGenerator(7));

thread1.start();

Thread thread2 = new Thread(new FibonacciGenerator(-10));

thread2.start();

Thread thread3 = new Thread(new FibonacciGenerator(0));

thread3.start();

}

}

1. **Generate a Java code to find the sum of N numbers using array and throw ArrayIndexOutOfBoundsException when the loop variable beyond the size N.**

**Sample Input : 5**

**1 2 3 4 5**

**Sample Output : 15**

**Test Cases**

1. **4, 10**
2. **-10**
3. **0**
4. **EIGHT SEVEN**

**ANSWER:**

import java.util.Scanner;

public class SumOfNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

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

int N = scanner.nextInt();

int[] numbers = new int[N];

System.out.println("Enter " + N + " numbers:");

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

numbers[i] = scanner.nextInt();

}

int sum = calculateSum(numbers);

System.out.println("Sum of numbers: " + sum);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException: Loop variable beyond the size N.");

} catch (Exception e) {

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

} finally {

scanner.close();

}

}

public static int calculateSum(int[] numbers) {

int sum = 0;

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

sum += numbers[i];

}

return sum;

}

}

1. **Using the concepts of thread with implementing Runnable interface in Java to find whether a given number is prime or not.**

**Sample Input : 5**

**Sample Output : 5 is Prime**

**Sample Output : 15**

**Test Cases**

1. **4**
2. **-10**
3. **0**
4. **EIGHT SEVEN**
5. **11.48**

**ANSWER:**

class PrimeChecker implements Runnable {

private final int number;

public PrimeChecker(int number) {

this.number = number;

}

public void run() {

if (isPrime(number)) {

System.out.println(number + " is Prime");

} else {

System.out.println(number + " is not Prime");

}

}

private boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

}

public class PrimeThreadExample {

public static void main(String[] args) {

Thread thread1 = new Thread(new PrimeChecker(4));

thread1.start();

Thread thread2 = new Thread(new PrimeChecker(-10));

thread2.start();

Thread thread3 = new Thread(new PrimeChecker(0));

thread3.start();

}

}

1. **Generate a Java code to handle Exceptions such as Arithmetic Exception, ArrayIndexOutOfBoundsException, NullPointerException using Multi-Catch Statements.**

**ANSWER:**

import java.util.Scanner;

public class MultiCatchExample {

public static void main(String[ ] args) {

Scanner scanner = new Scanner(System.in);

try {

int result = divideNumbers(10, 0);

System.out.println("Result of division: " + result);

int[] array = {1, 2, 3};

accessArrayElement(array, 5);

String str = null;

printStringLength(str);

} catch (ArithmeticException | ArrayIndexOutOfBoundsException | NullPointerException e) {

System.out.println("Exception caught: " + e.getClass().getSimpleName());

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

} catch (Exception e) {

System.out.println("Generic Exception caught: " + e.getClass().getSimpleName());

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

} finally {

scanner.close();

}

}

private static int divideNumbers(int numerator, int denominator) {

return numerator / denominator;

}

private static void accessArrayElement(int[ ] array, int index) {

System.out.println("Array element at index " + index + ": " + array[index]);

}

private static void printStringLength(String str) {

System.out.println("Length of the string: " + str.length());

}

}

1. **Generate a Java Code to Write and Read the string “Computer Science and Engineering” using FileWriter and FileReader Class.**

**ANSWER:**

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileReadWriteExample {

public static void main(String[] args) {

String filePath = "example.txt";

writeToFile(filePath, "Computer Science and Engineering");

String content = readFromFile(filePath);

System.out.println("Content read from file: " + content);

}

private static void writeToFile(String filePath, String content) {

try (FileWriter fileWriter = new FileWriter(filePath)) {

fileWriter.write(content);

System.out.println("Content written to file successfully.");

} catch (IOException e) {

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

}

}

private static String readFromFile(String filePath) {

StringBuilder content = new StringBuilder();

try (FileReader fileReader = new FileReader(filePath)) {

int character;

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

content.append((char) character);

}

System.out.println("Content read from file successfully.");

} catch (IOException e) {

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

}

return content.toString();

}

}

**24.Create a java program to construct the volume of Box using default constructor method.**

**ANSWER:**

class Box {

double length;

double width;

double height;

public Box() {

length = 1.0;

width = 1.0;

height = 1.0;

}

public double calculateVolume() {

return length \* width \* height;

}

}

public class BoxVolumeExample {

public static void main(String[] args) {

Box myBox = new Box();

System.out.println("Default values - Length: " + myBox.length +

", Width: " + myBox.width +

", Height: " + myBox.height);

double volume = myBox.calculateVolume();

System.out.println("Volume of the box: " + volume);

}

}

**25.Accept the string “Welcome to Saveetha university” from the user and perform the following operations by writing a suitable Java code.**

1. **Replace any word in the given String**
2. **Find the length**
3. **Uppercase Conversion**

**ANSWER:**

import java.util.Scanner;

public class StringOperationsExample {

public static void main(String[ ] args) {

Scanner scanner = new Scanner(System.in);

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

String inputString = scanner.nextLine();

String replacedString = inputString.replace("Saveetha", "XYZ");

int length = inputString.length();

String uppercaseString = inputString.toUpperCase();

System.out.println("i) Replaced String: " + replacedString);

System.out.println("ii) Length of the String: " + length);

System.out.println("iii) Uppercase Conversion: " + uppercaseString);

scanner.close();

}

}

**26.Create a HashTable to maintain a bank detail which includes Account number and Customer name. Let Account number be the key in the HashTable. Write a Java program to implement the following operations in the HashTable**

**i) Add 3 records**

**ii) Display the size of HashTable**

**iii) Clear the HashTable**

**ANSWER:**

import java.util.Hashtable;

import java.util.Map;

public class BankDetails {

public static void main(String[] args) {

Hashtable<Integer, String> bankTable = new Hashtable<>();

addRecord(bankTable, 101, "John Doe");

addRecord(bankTable, 102, "Jane Smith");

addRecord(bankTable, 103, "Bob Johnson");

displaySize(bankTable);

clearHashTable(bankTable);

}

private static void addRecord(Hashtable<Integer, String> bankTable, int accountNumber, String customerName) {

bankTable.put(accountNumber, customerName);

System.out.println("Record added - Account Number: " + accountNumber + ", Customer Name: " + customerName);

}

private static void displaySize(Hashtable<Integer, String> bankTable) {

System.out.println("Size of HashTable: " + bankTable.size());

}

private static void clearHashTable(Hashtable<Integer, String> bankTable) {

bankTable.clear();

System.out.println("HashTable cleared. Size now: " + bankTable.size());

}

}

**27.Create a employee record using map interface and do the following operations.**

**i. Add object iii. Remove specified object**

**ii. isEmpty or not iv. Clear**

**ANSWER:**

import java.util.HashMap;

import java.util.Map;

public class EmployeeRecord {

public static void main(String[] args) {

Map<Integer, Employee> employeeMap = new HashMap<>();

Employee emp1 = new Employee(101, "John Doe", "john@example.com");

Employee emp2 = new Employee(102, "Jane Doe", "jane@example.com");

employeeMap.put(emp1.getEmployeeId(), emp1);

employeeMap.put(emp2.getEmployeeId(), emp2);

System.out.println("Is Employee Map empty? " + employeeMap.isEmpty());

int employeeToRemove = 101;

if (employeeMap.containsKey(employeeToRemove)) {

employeeMap.remove(employeeToRemove);

System.out.println("Employee with ID " + employeeToRemove + " removed successfully.");

} else {

System.out.println("Employee with ID " + employeeToRemove + " not found.");

}

employeeMap.clear();

System.out.println("Employee Map cleared.");

System.out.println("Is Employee Map empty now? " + employeeMap.isEmpty());

}

}

class Employee {

private int employeeId;

private String name;

private String email;

public Employee(int employeeId, String name, String email) {

this.employeeId = employeeId;

this.name = name;

this.email = email;

}

public int getEmployeeId() {

return employeeId;

}

public String getName() {

return name;

}

public String getEmail() {

return email;

}

}

**28.Create a simple generics class with type parameters for sorting values of different types.**

**ANSWER:**

import java.util.Arrays;

class GenericSort<T extends Comparable<T>> {

private T[] array;

public GenericSort(T[] array) {

this.array = array;

}

public void sort() {

Arrays.sort(array);

}

public void displaySortedArray() {

System.out.print("Sorted Array: ");

for (T element : array) {

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

}

System.out.println();

}

}

public class GenericSortExample {

public static void main(String[] args) {

Integer[] intArray = {5, 2, 8, 1, 7};

GenericSort<Integer> intSort = new GenericSort<>(intArray);

intSort.sort();

intSort.displaySortedArray();

String[] stringArray = {"apple", "orange", "banana", "grape"};

GenericSort<String> stringSort = new GenericSort<>(stringArray);

stringSort.sort();

stringSort.displaySortedArray();

Double[] doubleArray = {3.2, 1.8, 2.5, 5.7, 4.1};

GenericSort<Double> doubleSort = new GenericSort<>(doubleArray);

doubleSort.sort();

doubleSort.displaySortedArray();

}

}

**29.Develop a Java code to insert the following elements, using ListIterator to append + symbol in each element and print them in reverse order. {C, A, E, B, D, F}.**

**ANSWER:**

import java.util.ArrayList;

import java.util.List;

import java.util.ListIterator;

public class ListIteratorExample {

public static void main(String[] args) {

List<String> elements = new ArrayList<>();

elements.add("C");

elements.add("A");

elements.add("E");

elements.add("B");

elements.add("D");

elements.add("F");

ListIterator<String> iterator = elements.listIterator();

while (iterator.hasNext()) {

String currentElement = iterator.next();

iterator.set(currentElement + "+");

}

System.out.println("Elements in reverse order:");

while (iterator.hasPrevious()) {

System.out.print(iterator.previous() + " ");

}

}

}

**30.Generate a Java code to perform simple arithmetic operations and to throw Arithmetic Exception for Division-by-Zero.**

**ANSWER:**

import java.util.Scanner;

public class ArithmeticOperationsExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num1 = scanner.nextInt();

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

int num2 = scanner.nextInt();

try {

int sum = num1 + num2;

int difference = num1 - num2;

int product = num1 \* num2;

if (num2 == 0) {

throw new ArithmeticException("Division by zero is not allowed.");

}

int quotient = num1 / num2;

System.out.println("Sum: " + sum);

System.out.println("Difference: " + difference);

System.out.println("Product: " + product);

System.out.println("Quotient: " + quotient);

} catch (ArithmeticException e) {

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

}

scanner.close();

}

}

**31.Write a Java program to create three threads in parallel and display the natural numbers in orders using sleep() method.**

**ANSWER:**

class NaturalNumbersThread extends Thread {

private int start;

private int end;

public NaturalNumbersThread(int start, int end) {

this.start = start;

this.end = end;

}

public void run() {

for (int i = start; i <= end; i++) {

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

try {

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class ParallelThreadsExample {

public static void main(String[] args) {

NaturalNumbersThread thread1 = new NaturalNumbersThread(1, 5);

NaturalNumbersThread thread2 = new NaturalNumbersThread(6, 10);

NaturalNumbersThread thread3 = new NaturalNumbersThread(11, 15);

thread1.start();

thread2.start();

thread3.start();

}

}

**32.If n = 8, then array ‘a’ will have 7 elements in the range from 1 to 8. For example {1, 4, 5, 3, 7, 8, 6}. One number will be missing in ‘a’ (2 in this case). Write a source code to find out that missing number**

**ANSWER**:

import java.util.Scanner;

public class MissingNumberExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the value of n: ");

int n = scanner.nextInt();

System.out.println("Enter the array 'a' with one missing number:");

int[] a = new int[n - 1];

for (int i = 0; i < n - 1; i++) {

a[i] = scanner.nextInt();

}

int missingNumber = findMissingNumber(n, a);

System.out.println("The missing number is: " + missingNumber);

scanner.close();

}

private static int findMissingNumber(int n, int[] a) {

int expectedSum = n \* (n + 1) / 2;

int actualSum = 0;

for (int num : a) {

actualSum += num;

}

return expectedSum - actualSum;

}

}

**33.Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call  
1 - method of parent class by object of parent class  
2 - method of child class by object of child class  
3 - method of parent class by object of child class**

**ANSWER:**

class ParentClass {

public void printParentMethod() {

System.out.println("This is parent class");

}

}

class ChildClass extends ParentClass {

public void printChildMethod() {

System.out.println("This is child class");

}

}

public class Main {

public static void main(String[] args) {

ParentClass parentObject = new ParentClass();

parentObject.printParentMethod();

ChildClass childObject = new ChildClass();

childObject.printChildMethod();

childObject.printParentMethod();

}

}

**34.Write a Java program to create a class Student and create constructor which assigns the values for the student details such as student name, register number, and five subject marks. Calculate the total and average of five subject marks and display the marks and average.**

**ANSWER:**

import java.util.Scanner;

class Student {

private String name;

private int registerNumber;

private int[] subjectMarks = new int[5];

public Student(String name, int registerNumber, int[] subjectMarks) {

this.name = name;

this.registerNumber = registerNumber;

this.subjectMarks = subjectMarks;

}

public int calculateTotalMarks() {

int total = 0;

for (int mark : subjectMarks) {

total += mark;

}

return total;

}

public double calculateAverageMarks() {

int totalMarks = calculateTotalMarks();

return (double) totalMarks / subjectMarks.length;

}

public void displayStudentDetails() {

System.out.println("Student Details:");

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

System.out.println("Register Number: " + registerNumber);

System.out.println("Subject Marks:");

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

System.out.println("Subject " + (i + 1) + ": " + subjectMarks[i]);

}

System.out.println("Total Marks: " + calculateTotalMarks());

System.out.println("Average Marks: " + calculateAverageMarks());

}

}

public class StudentDetailsExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter student name: ");

String name = scanner.nextLine();

System.out.print("Enter register number: ");

int registerNumber = scanner.nextInt();

int[] subjectMarks = new int[5];

System.out.println("Enter subject marks:");

for (int i = 0; i < 5; i++) {

System.out.print("Subject " + (i + 1) + ": ");

subjectMarks[i] = scanner.nextInt();

}

Student student = new Student(name, registerNumber, subjectMarks);

student.displayStudentDetails();

scanner.close();

}

}

**35.Generate a code to Count the Number of Words, Character and Lines from the File using Stream I/O in Java**

**ANSWER:**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class FileStatsExample {

public static void main(String[] args) {

String filePath = "your\_file\_path.txt";

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

int charCount = 0;

int wordCount = 0;

int lineCount = 0;

String line;

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

charCount += line.length();

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

wordCount += words.length;

lineCount++;

}

System.out.println("Number of Characters: " + charCount);

System.out.println("Number of Words: " + wordCount);

System.out.println("Number of Lines: " + lineCount);

} catch (IOException e) {

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

}

}

}

**36.Generate a code to non-negative integer’s num1 and num2 represented as strings; return the product of num1 and num2, also represented as a string.**

**ANSWER:**

public class MultiplyStrings {

public static String multiply(String num1, String num2) {

int m = num1.length();

int n = num2.length();

int[] result = new int[m + n];

for (int i = m - 1; i >= 0; i--) {

for (int j = n - 1; j >= 0; j--) {

int product = (num1.charAt(i) - '0') \* (num2.charAt(j) - '0');

int sum = product + result[i + j + 1];

result[i + j] += sum / 10;

result[i + j + 1] = sum % 10;

}

}

StringBuilder sb = new StringBuilder();

for (int digit : result) {

if (sb.length() != 0 || digit != 0) {

sb.append(digit);

}

}

return sb.length() == 0 ? "0" : sb.toString();

}

public static void main(String[] args) {

String num1 = "123";

String num2 = "456";

String product = multiply(num1, num2);

System.out.println("Product of " + num1 + " and " + num2 + " is: " + product);

}

}

**37.Implement pow(x, n), which calculates x raised to the power n**

**Input: x = 2.00000, n = 10**

**Output: 1024.00000**

**ANSWER:**

public class PowerCalculator {

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

if (n == 0) {

return 1.0;

}

double halfPow = myPow(x, n / 2);

if (n % 2 == 0) {

return halfPow \* halfPow;

} else {

return (n > 0) ? x \* halfPow \* halfPow : (1 / x) \* halfPow \* halfPow;

}

}

public static void main(String[] args) {

double x = 2.00000;

int n = 10;

double result = myPow(x, n);

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

}

}

**38.Given an integer array nums, find the subarray with the largest sum, and return its sum. Input: nums = [-2,1,-3,4,-1,2,1,-5,4]**

**Output: 6**

**Explanation: The subarray [4,-1, 2, 1] has the largest sum 6.**

**ANSWER:**

public class MaxSubarraySum {

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};

int result = maxSubArray(nums);

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

}

}

**39.Write a Java program which creates only one object. If user attempts to create second object, he should not be able to create it. (Using Exception Handling).**

**ANSWER:**

class SingletonObject {

private static SingletonObject instance;

private SingletonObject() {

}

public static SingletonObject getInstance() {

if (instance == null) {

instance = new SingletonObject();

return instance;

} else {

throw new RuntimeException("Cannot create more than one instance of SingletonObject");

}

}

}

public class SingletonExample {

public static void main(String[] args) {

try {

SingletonObject obj1 = SingletonObject.getInstance();

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

} catch (RuntimeException e) {

System.out.println(e.getMessage());}

}

}

**40.There is an exam room with n seats in a single row labeled from 0 to n - 1.When a student enters the room, they must sit in the seat that maximizes the distance to the closest person. If there are multiple such seats, they sit in the seat with the lowest number. If no one is in the room, then the student sits at seat number 0.Design a class that simulates the mentioned exam room. Implement the ExamRoom class: ExamRoom (int n) Initializes the object of the exam room with the number of the seats n. int seat () Returns the label of the seat at which the next student will set. Void leave (int p) indicates that the student sitting at seat p will leave the room. It is guaranteed that there will be a student sitting at seat p.**

**Input["ExamRoom", "seat", "seat", "seat", "seat", "leave", "seat"]**

**[[10], [], [], [], [], [4], []]**

**Output**

**[null, 0, 9, 4, 2, null, 5]**

**ANSWER:**

import java.util.TreeSet;

class ExamRoom {

private int n;

private TreeSet<Integer> occupiedSeats;

public ExamRoom(int n) {

this.n = n;

this.occupiedSeats = new TreeSet<>();

}

public int seat() {

if (occupiedSeats.isEmpty()) {

occupiedSeats.add(0);

return 0;

}

int maxDistance = occupiedSeats.first();

int seat = 0;

Integer prev = null;

for (int current : occupiedSeats) {

if (prev != null) {

int distance = (current - prev) / 2;

if (distance > maxDistance) {

maxDistance = distance;

seat = prev + distance;

}

}

prev = current;

}

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

seat = n - 1;

}

occupiedSeats.add(seat);

return seat;

}

public void leave(int p) {

occupiedSeats.remove(p);

}

public static void main(String[] args) {

ExamRoom examRoom = new ExamRoom(10);

System.out.println(examRoom.seat());

System.out.println(examRoom.seat());

System.out.println(examRoom.seat());

System.out.println(examRoom.seat());

examRoom.leave(4);

System.out.println(examRoom.seat());

}

}

**41.You have n tiles, where each tile has one letter tiles[i] printed on it. Return the number of possible non-empty sequences of letters you can make using the letters printed on those tiles.**

**Input: tiles = "AAB"**

**Output: 8**

**Explanation: The possible sequences are "A", "B", "AA", "AB", "BA", "AAB", "ABA", "BAA".**

**ANSWER:**

public class TileSequences {

public static int numTilePossibilities(String tiles) {

int[] charCount = new int[26];

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

charCount[c - 'A']++;

}

return backtrack(charCount);

}

private static int backtrack(int[] charCount) {

int count = 0;

for (int i = 0; i < 26; i++) {

if (charCount[i] > 0) {

charCount[i]--;

count++;

count += backtrack(charCount);

charCount[i]++; }

}

return count;

}

public static void main(String[] args) {

String tiles = "AAB";

int result = numTilePossibilities(tiles);

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

}

}

**42.Write a program to read the data from the file and copy it on another file**

**ANSWER:**

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileCopy {

public static void main(String[] args) {

String sourceFilePath = "source.txt";

String destinationFilePath = "destination.txt";

try {

copyFile(sourceFilePath, destinationFilePath);

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

} catch (IOException e) {

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

}

}

private static void copyFile(String sourcePath, String destinationPath) throws IOException {

try (FileReader reader = new FileReader(sourcePath);

FileWriter writer = new FileWriter(destinationPath)) {

char[] buffer = new char[1024];

int bytesRead;

while ((bytesRead = reader.read(buffer)) != -1) {

writer.write(buffer, 0, bytesRead);

}

}

}

}

**43.Implementing FileReader and FileWriter class to read and write the data from file**

**ANSWER:**

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileReadWriteExample {

public static void main(String[] args) {

String sourceFilePath = "source.txt";

String destinationFilePath = "destination.txt";

try {

String data = readFile(sourceFilePath);

writeFile(destinationFilePath, data);

System.out.println("Data copied from source to destination successfully.");

} catch (IOException e) {

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

}

}

private static String readFile(String filePath) throws IOException {

StringBuilder content = new StringBuilder();

try (FileReader reader = new FileReader(filePath)) {

int character;

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

content.append((char) character);

}

}

return content.toString();

}

private static void writeFile(String filePath, String data) throws IOException {

try (FileWriter writer = new FileWriter(filePath)) {

writer.write(data);

}

}

}

**44.Write a program to read the file using BufferedReader class**

**ANSWER:**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ReadFileWithBufferedReader {

public static void main(String[] args) {

String filePath = "sample.txt";

try {

readFromFile(filePath);

} catch (IOException e) {

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

}

}

private static void readFromFile(String filePath) throws IOException {

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

String line;

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

System.out.println(line);

}

}

}

}

**45.Write a program for counting number of characters, words and lines in a file**

**ANSWER:**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class FileStatistics {

public static void main(String[] args) {

String filePath = "sample.txt";

try {

FileStats stats = countStatistics(filePath);

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

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

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

} catch (IOException e) {

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

}

}

private static FileStats countStatistics(String filePath) throws IOException {

int charCount = 0;

int wordCount = 0;

int lineCount = 0;

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

String line;

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

lineCount++;

charCount += line.length();

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

wordCount += words.length;

}

}

return new FileStats(charCount, wordCount, lineCount);

}

private static class FileStats {

private final int charCount;

private final int wordCount;

private final int lineCount;

public FileStats(int charCount, int wordCount, int lineCount) {

this.charCount = charCount;

this.wordCount = wordCount;

this.lineCount = lineCount;

}

public int getCharCount() {

return charCount;

}

public int getWordCount() {

return wordCount;

}

public int getLineCount() {

return lineCount;

}

}

}

**46.Write a program in Java to input an NxN matrix and display it row-wise and column-wise**

**ANSWER:**

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];

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();

}

}

System.out.println("Matrix Display (Row-wise):");

displayMatrixRowWise(matrix);

System.out.println("\nMatrix Display (Column-wise):");

displayMatrixColumnWise(matrix);

scanner.close();

}

private static void displayMatrixRowWise(int[][] matrix) {

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

for (int j = 0; j < matrix[i].length; j++) {

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

}

System.out.println();

}

}

private static void displayMatrixColumnWise(int[][] matrix) {

for (int j = 0; j < matrix[0].length; j++) {

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

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

}

System.out.println();

}

}

}

**47.Write a java program which creates an interface IterF1 having 2 methods add () and sub (). Create a class which overloads the given methods for addition and subtraction of two numbers respectively.**

**ANSWER:**

interface IterF1 {

void add(int num1, int num2);

void sub(int num1, int num2);

}

class MathOperations implements IterF1 {

public void add(int num1, int num2) {

int sum = num1 + num2;

System.out.println("Sum: " + sum);

}

public void sub(int num1, int num2) {

int difference = num1 - num2;

System.out.println("Difference: " + difference);

}

}

public class InterfaceExample {

public static void main(String[] args) {

MathOperations mathObj = new MathOperations();

mathObj.add(10, 5);

mathObj.sub(10, 5);

}

}

**48.Write a Java program to calculate the rate of interest of the employee’s Provident Fund use the try and catch and finally block.**

**ANSWER:**

import java.util.Scanner;

public class ProvidentFundCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter the employee's basic salary: ");

double basicSalary = scanner.nextDouble();

System.out.print("Enter the number of years the employee has contributed to PF: ");

int yearsContributed = scanner.nextInt();

double interestRate = calculateInterestRate(basicSalary, yearsContributed);

System.out.println("The Provident Fund interest rate is: " + interestRate + "%");

} catch (Exception e) {

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

} finally {

scanner.close();

}

}

private static double calculateInterestRate(double basicSalary, int yearsContributed) {

if (basicSalary < 0 || yearsContributed < 0) {

throw new IllegalArgumentException("Invalid input. Basic salary and years contributed should be non-negative.");

}

double interestRate;

if (yearsContributed < 5) {

interestRate = 5.0;

} else if (yearsContributed >= 5 && yearsContributed < 10) {

interestRate = 7.5;

} else {

interestRate = 10.0;

}

return interestRate;

}

}

**49.Write a program to create a class MyThread in this class a constructor, call the base class constructor, using super and starts the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently**

**ANSWER:**

class MyThread extends Thread {

public MyThread() {

super("MyThread");

start();

}

public void run() {

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

System.out.println(Thread.currentThread().getName() + ": Count " + i);

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Thread interrupted.");

}

}

System.out.println(Thread.currentThread().getName() + ": Thread finished.");

}

}

public class ThreadExample {

public static void main(String[] args) {

System.out.println("Main thread started.");

MyThread myThread = new MyThread();

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

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

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("Main thread interrupted.");

}

}

System.out.println("Main thread finished.");

}

}

**50.Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". Define the two exception classes.**

**ANSWER:**

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]+$")) {

throw new NameNotValidException("Name should only contain letters.");

}

this.rollNo = rollNo;

this.name = name;

this.age = age;

this.course = course;

}

public void displayStudentInfo() {

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

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

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

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

}

}

public class StudentExceptionExample {

public static void main(String[ ] args) {

try {

Student validStudent = new Student(101, "JohnDoe", 18, "Computer Science");

validStudent.displayStudentInfo();

"Electrical Engineering");

"Mechanical Engineering");

}

catch (AgeNotWithinRangeException | NameNotValidException e) {

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

}

}

}