1. **Write a Java Program to Convert a Given Number of Days in Terms of Years, Weeks & Days.**

**Sample Input&Output::**

**Enter the number of days:756**

**No. of years:2**

**No. of weeks:3**

**No. of days:5**

**Test cases:**

1. **38**
2. **3.6**
3. **0**
4. **-365**
5. **-45**

import java.util.\*;

public class DaysConverter {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int totalDays = sc.nextInt();

if (totalDays < 0) {

System.out.println("Please enter a non-negative number of days.");

}

int years = totalDays / 365;

int remainingDays = totalDays % 365;

int weeks = remainingDays / 7;

int days = remainingDays % 7;

System.out.println("No. of years: " + years);

System.out.println("No. of weeks: " + weeks);

System.out.println("No. of days: " + days);

}

}

1. **Write a program to find the number of student users in the college, get the total users, staff users details from the client. Note for every 3 staff user there is one Non teaching staff user assigned by default.**

**Sample Input:**

**Total Users: 856**

**Staff Users: 126**

**Sample Output:**

**Student Users: 688**

**Test Cases:**

1. **Total User: 0**
2. **Total User: -143**
3. **Total User: 1026, Staff User: 1026**
4. **Total User: 450, Staff User: 540**
5. **Total User: 600, Staff User: 450**

import java.util.\*;

public class CollegeUserCount {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Total Users: ");

int totalUsers = sc.nextInt();

if (totalUsers < 0) {

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

}

System.out.print("Staff Users: ");

int staffUsers = sc.nextInt();

if (staffUsers < 0) {

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

}

int totUsers = totalUsers - staffUsers;

int nonTeachingStaff = staffUsers / 3;

System.out.println("total Users: " + totUsers);

System.out.println("Non-Teaching Staff Users: " + nonTeachingStaff);

int studentUsers=totUsers-nonTeachingStaff;

System.out.println("Student Users:"+studentUsers);

}

}

1. **Write a program to print number of factors and to print nth factor of the given number.**

**Sample Input:**

**Given Number: 100**

**N = 4**

**Sample Output:**

**Number of factors = 9**

**4th factor of 100 = 5**

**Test Cases:**

1. **Given Number = 512 , N = 6**
2. **Given Number = 343 , N = 7**
3. **Given Number = 1024 , N = 0**
4. **Given Number = -6561 , N = 3**
5. **Given Number = 0 , N = 2**

import java.util.Scanner;

public class Factors {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

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

int givenNumber = scanner.nextInt();

System.out.print("N = ");

int n = scanner.nextInt();

if (n <= 0) {

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

return;

}

int numberOfFactors = countFactors(givenNumber);

int nthFactor = findNthFactor(givenNumber, n);

System.out.println("Number of factors = " + numberOfFactors);

System.out.println(n + "th factor of " + givenNumber + " = " + nthFactor);

} catch (Exception e) {

System.out.println("Error: Invalid input. Please enter valid numbers.");

} finally {

scanner.close();

}

}

private static int countFactors(int number) {

int count = 0;

for (int i = 1; i <= Math.sqrt(number); i++) {

if (number % i == 0) {

count += (number / i == i) ? 1 : 2;

}

}

return count;

}

private static int findNthFactor(int number, int n) {

int count = 0;

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

if (number % i == 0) {

count++;

if (count == n) {

return i;

}

}

}

return -1;

}

}

1. **Write a program to print n prime numbers after nth Prime number**

**Sample Input:**

**N = 3**

**Sample Output:**

**3rd Prime number is 5**

**3 prime numbers after 5 are: 7, 11, 13**

**Test cases:**

1. **N = P**
2. **N = 0**
3. **N = -4**
4. **N = 11**
5. **N = 7.2**

import java.util.Scanner;

public class NthPrimeNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("N = ");

int n = scanner.nextInt();

if (n <= 0) {

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

return;

}

int nthPrime = findNthPrime(n);

int[] nextPrimes = findNextPrimes(nthPrime, n);

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

System.out.print(n + " prime numbers after " + nthPrime + " are: ");

for (int prime : nextPrimes) {

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

}

} catch (Exception e) {

System.out.println("Error: Invalid input. Please enter a valid number for N.");

} finally {

scanner.close();

}

}

private static boolean isPrime(int number) {

if (number <= 1) {

return false;

}

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

if (number % i == 0) {

return false;

}

}

return true;

}

private static int findNthPrime(int n) {

int count = 0;

int number = 1;

while (count < n) {

number++;

if (isPrime(number)) {

count++;

}

}

return number;

}

private static int[] findNextPrimes(int start, int n) {

int[] primes = new int[n];

int count = 0;

int number = start + 1;

while (count < n) {

if (isPrime(number)) {

primes[count] = number;

count++;

}

number++;

}

return primes;

}

}

1. **Write a Program to create a list of all numbers in a range which are perfect squares and the sum of the digits of the number is less than 10.**

**Sample Input & Output:**

**Enter lower range: 1**

**Enter upper range: 40**

**[1, 4, 9, 16, 25, 36]**

**Test case:**

1. **Enter lower range: 50**

**Enter upper range: 100**

1. **Enter lower range: 5**

**Enter upper range: 8**

1. **Enter lower range: 10**

**Enter upper range: 5**

1. **Enter lower range: 500**

**Enter upper range: 500**

1. **Enter lower range: 0**

**Enter upper range: -100**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class PerfectSquareList {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter lower range: ");

int lowerRange = scanner.nextInt();

System.out.print("Enter upper range: ");

int upperRange = scanner.nextInt();

if (lowerRange > upperRange) {

System.out.println("Invalid input: Lower range should be less than or equal to upper range.");

return;

}

List<Integer> resultList = new ArrayList<>();

for (int i = lowerRange; i <= upperRange; i++) {

if (isPerfectSquare(i) && digitSum(i) < 10) {

resultList.add(i);

}

}

System.out.println(resultList);

scanner.close();

}

private static boolean isPerfectSquare(int num) {

int sqrt = (int) Math.sqrt(num);

return sqrt \* sqrt == num;

}

private static int digitSum(int num) {

int sum = 0;

while (num > 0) {

sum += num % 10;

num /= 10;

}

return sum;

}

}

1. **Write a program to print unique permutations of a given number**

**Sample Input:**

**Given Number: 143**

**Sample Output:**

**Permutations are:**

**134**

**143**

**314**

**341**

**413**

**431**

**Test cases:**

1. **0**
2. **111**
3. **505**
4. **-143**
5. **-598**

import java.util.HashSet;

import java.util.Set;

import java.util.Scanner;

public class UniquePermutations {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

generateAndPrintPermutations(scanner.nextInt());

scanner.close();

}

private static void generateAndPrintPermutations(int number) {

System.out.println("Given Number: " + number);

Set<String> permutationsSet = new HashSet<>();

generatePermutations("", Integer.toString(Math.abs(number)), permutationsSet);

System.out.println("Permutations are:\n" + String.join("\n", permutationsSet) + "\n");

}

private static void generatePermutations(String prefix, String remaining, Set<String> permutationsSet) {

int n = remaining.length();

if (n == 0) permutationsSet.add(prefix);

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

generatePermutations(prefix + remaining.charAt(i), remaining.substring(0, i) + remaining.substring(i + 1, n), permutationsSet);

}

}

**7.Write a Program to create an array with the First Element as the Number and Second Element as the Square of the Number.**

**Sample Input:**

**Enter the lower range:45**

**Enter the upper range:49**

**Sample Output:**

**[(45, 2025), (46, 2116), (47, 2209), (48, 2304), (49, 2401)]**

**Test case:**

1. **Enter lower range: 50**

**Enter upper range: 100**

1. **Enter lower range: 5**

**Enter upper range: 8**

1. **Enter lower range: 10**

**Enter upper range: 5**

1. **Enter lower range: 500**

**Enter upper range: 500**

1. **Enter lower range: 0**

**Enter upper range: -100**

import java.util.Scanner;

public class NumberAndSquareArray {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the lower range: ");

int lr = scanner.nextInt();

System.out.print("Enter the upper range: ");

int ur = scanner.nextInt();

if(lr>ur)

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

for(int i=lr;i<=ur;i++){

if(i>0)

System.out.print("{"+i+","+i\*i+"}");

}

}

}

**8.Develop a JAVA code to display the balance. Include the following members:**

* **Design a class to represent a bank account.**
* **Data Members: Name of the depositor, Account number, Type of account(Savings/Current), Balance amount in the account(Minimum balance is Rs.500.00)**
* **Methods:**

1. **To read account number, Depositor name, Type of account.**
2. **To deposit an amount (Deposited amount should be added with it)**
3. **To withdraw an amount after checking balance(Minimum balance must be Rs.500.00**

**Note : Assume that balance amount = 10000**

**Test Cases**

1. **100, Raja, S, 8000**
2. **Raja, 100, S, 9000**
3. **101, Rani, S, 12000**
4. **102, Ragu, W, 8000**
5. **103, Ravi, C, 10000**

import java.util.Scanner;

class BankAccount {

private String depositorName;

private int accountNumber;

private char accountType;

private double balance;

public BankAccount(int accountNumber, String depositorName, char accountType, double initialBalance) {

this.accountNumber = accountNumber;

this.depositorName = depositorName;

this.accountType = accountType;

this.balance = initialBalance;

}

public void displayBalance() {

System.out.println("Account Number: " + accountNumber);

System.out.println("Depositor Name: " + depositorName);

System.out.println("Account Type: " + accountType);

System.out.println("Balance: Rs." + balance);

}

public void deposit(double amount) {

balance += amount;

System.out.println("Deposit of Rs." + amount + " successful.");

displayBalance();

}

public void withdraw(double amount) {

if (balance - amount >= 500) {

balance -= amount;

System.out.println("Withdrawal of Rs." + amount + " successful.");

} else {

System.out.println("Insufficient balance. Minimum balance of Rs.500 must be maintained.");

}

displayBalance();

}

}

class TestBankAccount {

public static void main(String[] args) {

BankAccount account1 = new BankAccount(100, "Raja", 'S', 8000);

account1.displayBalance();

account1.deposit(1000);

account1.withdraw(2000);

BankAccount account2 = new BankAccount(101, "Rani", 'S', 12000);

account2.displayBalance();

BankAccount account3 = new BankAccount(102, "Ragu", 'W', 8000);

account3.withdraw(3000);

BankAccount account4 = new BankAccount(103, "Ravi", 'C', 10000);

account4.displayBalance();

}

}

**9. Develop a code to Reverse and Add a Number until you get a Palindrome.**

**Sample Input If 7325 is input number, then**

**7325 (Input Number) + 5237 (Reverse Of Input Number) = 12562**

**12562 + 26521 = 39083**

**39083 + 38093 = 77176**

**77176 + 67177 = 144353**

**144353 + 353441 = 497794 (Palindrome)**

**Test Cases**

1. **8765**
2. **-8765**
3. **0**
4. **EIGHT FIVE**

**5.87.57**

import java.util.Scanner;

public class ReverseAndAddPalindrome {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

if (scanner.hasNextInt()) {

int inputNumber = scanner.nextInt();

if (inputNumber < 0) {

System.out.println("Input should be a non-negative integer.");

} else {

int steps = reverseAndAddToPalindrome(inputNumber);

System.out.println("Palindrome obtained in " + steps + " steps.");

}

} else {

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

}

scanner.close();

}

private static int reverseAndAddToPalindrome(int number) {

int steps = 0;

while (!isPalindrome(number)) {

int reverse = reverseNumber(number);

number += reverse;

System.out.println(number);

steps++;

}

return steps;

}

private static boolean isPalindrome(int number) {

int original = number;

int reverse = reverseNumber(number);

return original == reverse;

}

private static int reverseNumber(int number) {

int reverse = 0;

while (number != 0) {

int digit = number % 10;

reverse = reverse \* 10 + digit;

number /= 10;

}

return reverse;

}

}

**10. Create Customer class with deposit() and withdraw() as synchronized methods. Declare AccountNo, AccName and Balance as Instance Variables inside the class. From the main class, Input the amount for withdraw() operation and if requested amount is not available in existing Balance amount, withdraw() method should be temporarily suspended using wait() method until deposit() method receives the input for amount, to be added in the existing Balance amount and then withdraw() would be completed in a successful manner. Develop the above scenario using Synchronization and Inter-Thread Communication.**

**Note : existing Bank balance amount 10000**

**Sample Input : 12000, 3000**

**Sample Output : Withdraw operation success, balance amount 1000**

**Test Cases**

1. **11000, 4000**
2. **-10000, -2000**
3. **0, 0**
4. **EIGHT SEVEN, FIVE**

**5.100.67, 200.68**

class Customer {

private int accountNo;

private String accName;

private double balance = 10000;

public Customer(int accountNo, String accName) {

this.accountNo = accountNo;

this.accName = accName;

}

public synchronized void deposit(double amount) {

System.out.println("Deposit request received: Rs." + amount);

balance += amount;

System.out.println("Deposit successful. Updated balance: Rs." + balance);

notify();

}

public synchronized void withdraw(double amount) {

System.out.println("Withdrawal request received: Rs." + amount);

if (amount > balance) {

System.out.println("Insufficient funds. Waiting for deposit...");

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

balance -= amount;

System.out.println("Withdrawal successful. Updated balance: Rs." + balance);

}

}

class TestCustomer {

public static void main(String[] args) {

Customer customer = new Customer(123, "John");

new Thread(() -> customer.withdraw(12000)).start();

new Thread(() -> customer.deposit(3000)).start();

new Thread(() -> customer.withdraw(-10000)).start();

new Thread(() -> customer.deposit(-2000)).start();

new Thread(() -> customer.withdraw(0)).start();

new Thread(() -> customer.deposit(0)).start();

new Thread(() -> customer.withdraw(0)).start();

new Thread(() -> customer.deposit(0)).start();

}

}

* 1. **Given an integer n, return a string array answer (1-indexed) where:**

**answer[i] == "FizzBuzz" if i is divisible by 3 and 5.**

**answer[i] == "Fizz" if i is divisible by 3.**

**answer[i] == "Buzz" if i is divisible by 5.**

**answer[i] == i (as a string) if none of the above conditions are true.**

**Example 1:**

Input: n = 3

Output: ["1","2","Fizz"]

**Test Case**

|  |  |
| --- | --- |
| **Test Case** | **Inputs** |
|  | n = 5 |
|  | n = 10 |
|  | n = 12 |
|  | n = 18 |
|  | n = 20 |

public class FizzBuzz {

public static String[] fizzBuzz(int n) {

String[] result = new String[n];

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

if (i % 3 == 0 && i % 5 == 0) {

result[i - 1] = "FizzBuzz";

} else if (i % 3 == 0) {

result[i - 1] = "Fizz";

} else if (i % 5 == 0) {

result[i - 1] = "Buzz";

} else {

result[i - 1] = String.valueOf(i);

}

}

return result;

}

public static void main(String[] args) {

int n = 15;

String[] result = fizzBuzz(n);

for (String s : result) {

System.out.println(s);

}

}

}

* 1. **Write a Java program to find the common elements in two array of Positive integer**

Sample Input:

[1, 2, 3, 4]

[2, 4, 5, 6, 7]

Expected output: [2, 4]

**Test Case**

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Inputs-1** | **Inputs-2** |
|  | [1, 2, 3, 4] | [4,5,6,7,8] |
|  | [a, b, c, d] | [a, b, c, d] |
|  | [1, -2, 3, 4] | [1,-2,5,7,8] |
|  | [@, #, 34, 45] | [@,#,%,$,] |
|  | [45,78,56,89] | [92,34,56,-78,-90] |

public class CommonCharacters {

public static void main(String[] args) {

char[] array1 = {'a', 'b', 'c', 'd', 'e'};

char[] array2 = {'b', 'd', 'f', 'g', 'h'};

System.out.println("Common characters:");

printCommonCharacters(array1, array2);

}

public static void printCommonCharacters(char[] array1, char[] array2) {

for (char ch : array1) {

for (char ch2 : array2) {

if (ch == ch2) {

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

break;

}

}

}

}

}

* 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 Case**

|  |  |
| --- | --- |
| **Test Case** | **Inputs-1** |
|  | **1,7,6,9,5** |
|  | **S, Q, l, K,7, j, M** |
|  | **M, j, L, &, @, G** |
|  | **D, K, I, 6, L, \*** |

import java.util.\*;

public class CharacterCount {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int uppercaseCount = 0, lowercaseCount = 0, numberCount = 0, specialCharCount = 0;

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

while (true) {

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

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

if (input == '\*') {

break;

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

uppercaseCount++;

} else if (Character.isLowerCase(input)) {

lowercaseCount++;

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

numberCount++;

} else {

specialCharCount++;

}

}

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

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

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

System.out.println("Total count of special characters: " + specialCharCount);

}

}

* 1. **Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.**

**Symbol Value**

**I 1**

**V 5**

**X 10**

**L 50**

**C 100**

**D 500**

**M 1000**

**For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.**

**Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:**

* **I can be placed before V (5) and X (10) to make 4 and 9.**
* **X can be placed before L (50) and C (100) to make 40 and 90.**
* **C can be placed before D (500) and M (1000) to make 400 and 900.**

**Given a roman numeral, convert it to an integer**.

**Example:**

**Input:** s = "III"

**Output:** 3

|  |  |
| --- | --- |
| **Test Case** | **Inputs** |
|  | LVIII |
|  | MCMXCI |
|  | V |
|  | LZAII |
|  | MCCDTIV |

import java.util.HashMap;

import java.util.Map;

import java.util.\*;

public class RomanToIntegerConverter {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter a Roman numeral: ");

String romanNumeral = sc.nextLine().toUpperCase();

int result = romanToInteger(romanNumeral);

if (result != -1) {

System.out.println("The equivalent integer for " + romanNumeral + " is: " + result);

} else {

System.out.println("Invalid Roman numeral entered.");

}

}

private static int romanToInteger(String roman) {

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

romanMap.put('I', 1);

romanMap.put('V', 5);

romanMap.put('X', 10);

romanMap.put('L', 50);

romanMap.put('C', 100);

romanMap.put('D', 500);

romanMap.put('M', 1000);

int result = 0;

int prevValue = 0;

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

int currentValue = romanMap.get(roman.charAt(i));

if (currentValue < prevValue) {

result -= currentValue;

} else {

result += currentValue;

}

prevValue = currentValue;

}

return result;

}

}

* 1. **Given two strings ransomNote and magazine, return true if ransomNote can be constructed by using the letters from magazine and false otherwise. Each letter in magazine can only be used once in ransomNote.**

**Example 1:**

Input: ransomNote = "a", magazine = "b"

Output: false

**Test Case**

|  |  |
| --- | --- |
| **Test Case** | **Inputs** |
|  | ransomNote = "aa", magazine = "ab" |
|  | ransomNote = "aa", magazine = "aab" |
|  | ransomNote = "abc", magazine = "abc" |
|  | ransomNote = "good", magazine = "better" |
|  | ransomNote = "xyz", magazine = "123" |

import java.util.Scanner;

public class SubstringCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter ransomeNote: ");

String stringA = scanner.nextLine();

System.out.print("Enter magazine: ");

String stringB = scanner.nextLine();

boolean isSubstring = isSubstring(stringA, stringB);

System.out.println("Is ransomNote constructed from magazine? " + isSubstring);

scanner.close();

}

public static boolean isSubstring(String A, String B) {

return B.indexOf(A) != -1;

}

}

* 1. **You are given an m x n binary matrix mat of 1's (representing soldiers) and 0's (representing civilians). The soldiers are positioned in front of the civilians. That is, all the 1's will appear to the left of all the 0's in each row.**

**A row i is weaker than a row j if one of the following is true:**

**The number of soldiers in row i is less than the number of soldiers in row j.**

**Both rows have the same number of soldiers and i < j. Return the indices of the k weakest rows in the matrix ordered from weakest to strongest.**

**Example 1:**

Input: mat =

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

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

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

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

[1,1,1,1,1]],

k = 3

Output: [2,0,3]

**Explanation:**

The number of soldiers in each row is:

- Row 0: 2

- Row 1: 4

- Row 2: 1

- Row 3: 2

- Row 4: 5

The rows ordered from weakest to strongest are [2,0,3,1,4].

**Example 2:**

Input: mat =

[[1, 0, 0, 0],

[1, 1, 1,1],

[1, 0, 0, 0],

[1, 0, 0,0]],

k = 2

Output: [0,2]

Explanation:

The number of soldiers in each row is:

- Row 0: 1

- Row 1: 4

- Row 2: 1

- Row 3: 1

The rows ordered from weakest to strongest are [0, 2, 3, 1].

import java.util.\*;

public class WeakestRows {

public static int[] kWeakestRows(int[][] mat, int k) {

PriorityQueue<int[]> heap = new PriorityQueue<>(Comparator.comparingInt(a -> a[0]));

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

heap.offer(new int[]{Arrays.stream(mat[i]).sum(), i});

int[] result = new int[k];

for (int i = 0; i < k; i++)

result[i] = heap.poll()[1];

return result;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int m = scanner.nextInt();

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

int n = scanner.nextInt();

int[][] mat = new int[m][n];

System.out.println("Enter the matrix elements (1 or 0): ");

for (int i = 0; i < m; i++)

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

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

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

int k = scanner.nextInt();

int[] result = kWeakestRows(mat, k);

System.out.println("The indices of the k weakest rows are: " + Arrays.toString(result));

}

}

* 1. **Given an integer num, return the number of steps to reduce it to zero. In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.**

**Example 1:**

Input: num = 14

Output: 6

Explanation:

Step 1) 14 is even; divide by 2 and obtain 7.

Step 2) 7 is odd; subtract 1 and obtain 6.

Step 3) 6 is even; divide by 2 and obtain 3.

Step 4) 3 is odd; subtract 1 and obtain 2.

Step 5) 2 is even; divide by 2 and obtain 1.

Step 6) 1 is odd; subtract 1 and obtain 0.

**Test Case**

|  |  |
| --- | --- |
| **Test Case** | **Inputs** |
|  | n = 5 |
|  | n = 10 |
|  | n = 12 |
|  | n = 18 |
|  | n = 20 |

import java.util.\*;

class reducetozero{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

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

int n=sc.nextInt();

int s=0;

while(n>0){

if(n%2==0){

n=n/2;

}

else{

n-=1;

}

s++;

}

System.out.println("Number of steps="+s);

}

}

* 1. **Develop a programme that uses Multiple Inheritance concepts to compute a student's grades in six subjects. The total and aggregate are then calculated, and the student's grade is displayed. If the student achieves an aggregate of more than 75%, the grade is Distinction. If the aggregate is between 60 and 75, the grade is First Division. If the aggregate is between 50 and 60, the grade is Second Division. If the aggregate is between 40 and 50, the grade is Third Division. Otherwise, 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

Enter the marks in Chemistry: 92

Enter the marks in Professional Ethics: 93

Total= 551

Aggregate = 91.83

Class: DISTINCTION

**Test Case**

|  |  |
| --- | --- |
| **Test Case** | **Inputs** |
|  | 18, 76,93,65,63,98 |
|  | 73,78,79,75,87,0 |
|  | 98,106,120,95,98,34 |
|  | 96,73, -85,95,84,98 |
|  | 78,59.8,76,79,97,67 |

import java.util.Scanner;

interface Marks {

void inputMarks();

}

interface Result extends Marks {

void calculateTotal();

void calculateAggregate();

void displayGrade();

}

class Student implements Result {

private int python, cProgramming, mathematics, physics, chemistry, professionalEthics;

private int total;

private double aggregate;

@Override

public void inputMarks() {

Scanner scanner = new Scanner(System.in);

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

python = scanner.nextInt();

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

cProgramming = scanner.nextInt();

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

mathematics = scanner.nextInt();

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

physics = scanner.nextInt();

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

chemistry = scanner.nextInt();

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

professionalEthics = scanner.nextInt();

}

@Override

public void calculateTotal() {

total = python + cProgramming + mathematics + physics + chemistry + professionalEthics;

}

@Override

public void calculateAggregate() {

aggregate = total / 6.0;

}

@Override

public void displayGrade() {

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

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

if (aggregate > 75) {

System.out.println("Class: DISTINCTION");

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

System.out.println("Class: FIRST DIVISION");

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

System.out.println("Class: SECOND DIVISION");

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

System.out.println("Class: THIRD DIVISION");

} else {

System.out.println("Class: FAIL");

}

}

}

class StudentGradeProgram {

public static void main(String[] args) {

Student student = new Student();

student.inputMarks();

student.calculateTotal();

student.calculateAggregate();

student.displayGrade();

}

}

* 1. **Write a program to calculate tax given the following conditions:**
  2. **If income is less than or equal to 2,50,000 then no tax**
  3. **If taxable income is 2,50,001 – 5,00,000 the charge 10% tax**
  4. **If taxable income is 5,00,001 – 10,00,000 the charge 20% tax**
  5. **If taxable income is above 10,00,001 then charge 30% tax**

**Sample Input:**

Enter the income: 600000

**Sample Output:**

Taxable Income: 350000

Tax= 35000

**Test Case**

|  |  |
| --- | --- |
| **Test Case** | **Inputs** |
|  | 400700 |
|  | 2789239 |
|  | 150000 |
|  | 00000 |
|  | -125486 |

import java.util.Scanner;

public class TaxCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double income = scanner.nextDouble();

double taxableIncome = calculateTaxableIncome(income);

double tax = calculateTax(taxableIncome);

System.out.println("Taxable Income: " + taxableIncome);

System.out.println("Tax= " + tax);

scanner.close();

}

public static double calculateTaxableIncome(double income) {

double taxableIncome = 0;

if (income > 250000 && income <= 500000) {

taxableIncome = income - 250000;

} else if (income > 500000 && income <= 1000000) {

taxableIncome = 250000 + (income - 500000);

} else if (income > 1000000) {

taxableIncome = 250000 + 500000 + (income - 1000000);

}

return taxableIncome;

}

public static double calculateTax(double taxableIncome) {

double tax = 0;

if (taxableIncome > 0 && taxableIncome <= 250000) {

tax = 0;

} else if (taxableIncome > 250000 && taxableIncome <= 500000) {

tax = 0.1 \* taxableIncome;

} else if (taxableIncome > 500000 && taxableIncome <= 1000000) {

tax = 0.2 \* taxableIncome;

} else if (taxableIncome > 1000000) {

tax = 0.3 \* taxableIncome;

}

return tax;

}

}