

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**SCENARIO BASED QUESTIONS-60**

# 

# SUBMITTED BY

BHAVADHARANI VEMBI K (192224172)

**SUBMITTED TO**

Dr.S.PADMAKALA

# COURSE CODE / COURSE NAME

**CSA0968/ PROGRAMMING IN JAVA WITH**

**COLLECTION INTERFACES**

**SLOT A**

1. **Write a Java program named AllEqual.java that takes three integer command-line arguments and prints "equal" if all three integers are equal, and "not equal" otherwise. Ensure the program handles edge cases such as non-integer inputs and the incorrect number of arguments gracefully.**

**Sample Input & Output: java 4 4 4 equal**

import java.util.Scanner;

public class Equal{

public static void main(String args[]){

int num1,num2,num3;

Scanner input=new Scanner(System.in);

System.out.println("Enter the three numbers : ");

num1=input.nextInt();

num2=input.nextInt();

num3=input.nextInt();

if(num1==num2 && num1==num3){

System.out.println("The given number are equal");

}else{

System.out.println("The given number are not equal");

}

}

}

1. **Bank Account Management System**

**Description: Write a program to manage bank accounts. Each account has an account number, a balance, and an account type (Savings or Checking). The program should allow depositing, withdrawing, and viewing the account balance. Savings accounts should earn interest monthly at a rate of 4%, while checking accounts have a minimum balance requirement of $500, and if the balance falls below this, a fee of $50 is charged**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class BankAccount {

protected String accountNumber;

protected double balance;

protected String accountType;

public BankAccount(String accountNumber, double balance, String accountType) {

this.accountNumber = accountNumber;

this.balance = balance;

this.accountType = accountType;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) {

balance -= amount;

}

public double getBalance() {

return balance;

}

}

class SavingsAccount extends BankAccount {

public SavingsAccount(String accountNumber, double balance) {

super(accountNumber, balance, "Savings");

}

public void addInterest() {

balance \*= 1.04;

}

}

class CheckingAccount extends BankAccount {

public CheckingAccount(String accountNumber, double balance) {

super(accountNumber, balance, "Checking");

}

@Override

public void withdraw(double amount) {

if (balance - amount < 500) {

balance -= amount + 50;

} else {

balance -= amount;

}

}

}

public class BankAccountManagementSystem {

private Map<String, BankAccount> accounts;

private Scanner scanner;

public BankAccountManagementSystem() {

accounts = new HashMap<>();

scanner = new Scanner(System.in);

}

public void createAccount() {

System.out.println("Enter account number:");

String accountNumber = scanner.nextLine();

System.out.println("Enter initial balance:");

double balance = scanner.nextDouble();

scanner.nextLine(); // consume newline left-over

System.out.println("Enter account type (Savings/Checking):");

String accountType = scanner.nextLine();

if (accountType.equalsIgnoreCase("Savings")) {

accounts.put(accountNumber, new SavingsAccount(accountNumber, balance));

} else if (accountType.equalsIgnoreCase("Checking")) {

accounts.put(accountNumber, new CheckingAccount(accountNumber, balance));

}

}

public void deposit() {

System.out.println("Enter account number:");

String accountNumber = scanner.nextLine();

System.out.println("Enter amount to deposit:");

double amount = scanner.nextDouble();

scanner.nextLine(); // consume newline left-over

if (accounts.containsKey(accountNumber)) {

accounts.get(accountNumber).deposit(amount);

}

}

public void withdraw() {

System.out.println("Enter account number:");

String accountNumber = scanner.nextLine();

System.out.println("Enter amount to withdraw:");

double amount = scanner.nextDouble();

scanner.nextLine(); // consume newline left-over

if (accounts.containsKey(accountNumber)) {

accounts.get(accountNumber).withdraw(amount);

}

}

public void viewBalance() {

System.out.println("Enter account number:");

String accountNumber = scanner.nextLine();

if (accounts.containsKey(accountNumber)) {

System.out.println("Balance: " + accounts.get(accountNumber).getBalance());

}

}

public void addInterest() {

for (BankAccount account : accounts.values()) {

if (account.accountType.equalsIgnoreCase("Savings")) {

((SavingsAccount) account).addInterest();

}

}

}

public void run() {

while (true) {

System.out.println("1. Create account");

System.out.println("2. Deposit");

System.out.println("3. Withdraw");

System.out.println("4. View balance");

System.out.println("5. Add interest");

System.out.println("6. Exit");

System.out.println("Enter choice:");

int choice = scanner.nextInt();

scanner.nextLine(); // consume newline left-over

switch (choice) {

case 1:

createAccount();

break;

case 2:

deposit();

break;

case 3:

withdraw();

break;

case 4:

viewBalance();

break;

case 5:

addInterest();

break;

case 6:

System.exit(0);

break;

default:

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

}

}

}

public static void main(String[] args) {

BankAccountManagementSystem system = new BankAccountManagementSystem();

system.run();

}

}

1. **Write a Java program to count the number of prime and composite numbers from a list of integers entered by the user. The program should handle edge cases such as negative numbers, non-integer inputs, and zero, and it should also handle an empty list gracefully.**

**Sample Input & Output: Enter numbers (separated by spaces): 2 3 4 5 6 7 8 9 10 Number of prime numbers: 4 Number of composite numbers:5**

import java.util.Scanner;

public class PrimeCompositeCounter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter numbers (separated by spaces):");

String input = scanner.nextLine();

String[] numbers = input.split(" ");

int primeCount = 0;

int compositeCount = 0;

for (String number : numbers) {

try {

int num = Integer.parseInt(number);

if (num < 0) {

System.out.println("Ignoring negative number: " + num);

continue;

}

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

compositeCount++;

} else if (isPrime(num)) {

primeCount++;

} else {

compositeCount++;

}

} catch (NumberFormatException e) {

System.out.println("Ignoring non-integer input: " + number);

}

}

System.out.println("Number of prime numbers: " + primeCount);

System.out.println("Number of composite numbers: " + compositeCount);

}

public static boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

}

1. **Write a Java program to find the Mth maximum number and Nth minimum number in an array. After identifying these numbers, calculate the sum and difference between them. Ensure the program handles edge cases, such as invalid M or N values, appropriately. Sample Input & Output: Enter the array elements: [3, 1, 4, 9, 2, 7, 6] Enter the value of M (for Mth maximum): 2 Enter the value of N (for Nth minimum): 3 Mth maximum number: 7 Nth minimum number: 3 Sum: 10 Difference: 4**

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the array elements (separated by spaces):");

String input = scanner.nextLine();

String[] numbers = input.split(" ");

int[] array = new int[numbers.length];

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

array[i] = Integer.parseInt(numbers[i]);

}

System.out.println("Enter the value of M (for Mth maximum):");

int m = scanner.nextInt();

System.out.println("Enter the value of N (for Nth minimum):");

int n = scanner.nextInt();

if (m < 1 || n < 1 || m > array.length || n > array.length) {

System.out.println("Invalid M or N value");

return;

}

Arrays.sort(array);

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

int nthMin = array[n - 1];

System.out.println("Mth maximum number: " + mthMax);

System.out.println("Nth minimum number: " + nthMin);

System.out.println("Sum: " + (mthMax + nthMin));

System.out.println("Difference: " + (mthMax - nthMin));

}

}

1. **Write a Java program to check if a given integer is a palindrome. An integer is a palindrome if it reads the same backward as forward.**

**• Input: A single integer. • Output: Print "Palindrome" if the number is a palindrome, otherwise print "Not a Palindrome"**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter an integer:");

int num = scanner.nextInt();

if (isPalindrome(num)) {

System.out.println("Palindrome");

} else {

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

}

}

public static boolean isPalindrome(int num) {

int reversed = 0;

int original = num;

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return original == reversed;

}

}

1. **Write a Java program to check if two given strings are anagrams of each other. An anagram of a string is another string that contains the same characters, only the order of characters can be different. • Input: Two strings. • Output: Print "Anagrams" if the strings are anagrams, otherwise print "Not Anagrams"**

import java.util.Arrays;

import java.util.Scanner;

public class AnagramChecker {

public static void main(String[] args) {

String str1;

String str2;

Scanner input=new Scanner(System.in);

System.out.println("Enter the two strings:");

str1=input.nextLine();

str2=input.nextLine();

if (isAnagram(str1, str2)) {

System.out.println(str1 + " and " + str2 + " are anagrams.");

} else {

System.out.println(str1 + " and " + str2 + " are not anagrams.");

}

}

public static boolean isAnagram(String str1, String str2) {

if (str1.length() != str2.length()) {

return false;

}

char[] charArray1 = str1.toLowerCase().toCharArray();

char[] charArray2 = str2.toLowerCase().toCharArray();

Arrays.sort(charArray1);

Arrays.sort(charArray2);

return Arrays.equals(charArray1, charArray2);

}

}

1. **Given an array containing n-1 distinct numbers taken from the range 1 to n, find the missing number. • Input: An array of integers. • Output: The missing integer.**

public class MissingNumber {

public static void main(String[] args) {

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

int n = arr.length + 1;

int missingNumber = findMissingNumber(arr, n);

System.out.println("Missing number: " + missingNumber);

}

public static int findMissingNumber(int[] arr, int n) {

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

int sumOfArray = 0;

for (int num : arr) {

sumOfArray += num;

}

return sumOfRange - sumOfArray;

}

}

1. **Write a Java program to multiply two matrices. The program should read two matrices and print their product. • Input: Two 2D arrays (matrices). • Output: The product matrix.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of rows for the first matrix:");

int rows1 = scanner.nextInt();

System.out.println("Enter the number of columns for the first matrix:");

int cols1 = scanner.nextInt();

int[][] matrix1 = new int[rows1][cols1];

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

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols1; j++) {

matrix1[i][j] = scanner.nextInt();

}

}

System.out.println("Enter the number of rows for the second matrix:");

int rows2 = scanner.nextInt();

System.out.println("Enter the number of columns for the second matrix:");

int cols2 = scanner.nextInt();

if (cols1 != rows2) {

System.out.println("Matrices cannot be multiplied");

return;

}

int[][] matrix2 = new int[rows2][cols2];

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

for (int i = 0; i < rows2; i++) {

for (int j = 0; j < cols2; j++) {

matrix2[i][j] = scanner.nextInt();

}

}

int[][] product = multiplyMatrices(matrix1, matrix2);

System.out.println("Product matrix:");

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

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

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

}

System.out.println();

}

}

public static int[][] multiplyMatrices(int[][] matrix1, int[][] matrix2) {

int rows1 = matrix1.length;

int cols1 = matrix1[0].length;

int cols2 = matrix2[0].length;

int[][] product = new int[rows1][cols2];

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols2; j++) {

for (int k = 0; k < cols1; k++) {

product[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

return product;

}

}

1. **Write a Java program to calculate the frequency of each character in a given string. • Input: A single string. • Output: Print each character and its frequency.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

int[] charCount = new int[256];

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

charCount[c]++;

}

for (int i = 0; i < 256; i++) {

if (charCount[i] > 0) {

System.out.println((char) i + ": " + charCount[i]);

}

}

}

}

1. **Implement the binary search algorithm to find the index of a target value within a sorted array. • Input: A sorted array and a target value. • Output: The index of the target value if found, otherwise -1.**

public class BinarySearch {

public static int binarySearch(int[] array, int target) {

int left = 0;

int right = array.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (array[mid] == target) {

return mid;

} else if (array[mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

public static void main(String[] args) {

int[] array = {1, 3, 5, 7, 9, 11, 13, 15};

int target = 9;

int result = binarySearch(array, target);

if (result != -1) {

System.out.println("Target found at index " + result);

} else {

System.out.println("Target not found");

}

}

}

1. **Write a Java program to check whether the parentheses in a given expression are balanced. • Input: A string containing parentheses. • Output: Print "Balanced" if the parentheses are balanced, otherwise print "Not Balanced"**

import java.util.Stack;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter an expression:");

String expression = scanner.nextLine();

if (isBalanced(expression)) {

System.out.println("Balanced");

} else {

System.out.println("Not Balanced");

}

}

public static boolean isBalanced(String expression) {

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

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

if (c == '(' || c == '[' || c == '{') {

stack.push(c);

} else if (c == ')' || c == ']' || c == '}') {

if (stack.isEmpty()) {

return false;

}

char opening = stack.pop();

if ((c == ')' && opening != '(') ||

(c == ']' && opening != '[') ||

(c == '}' && opening != '{')) {

return false;

}

}

}

return stack.isEmpty();

}

}

1. **Write a Java program to remove duplicates from a sorted array and return the new length of the array. • Input: A sorted array of integers. • Output: The new length of the array after removing duplicates**.

public class Main {

public static int removeDuplicates(int[] array) {

if (array.length == 0) {

return 0;

}

int writePointer = 1;

for (int readPointer = 1; readPointer < array.length; readPointer++) {

if (array[readPointer] != array[readPointer - 1]) {

array[writePointer] = array[readPointer];

writePointer++;

}

}

return writePointer;

}

public static void main(String[] args) {

int[] array = {1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4};

int newLength = removeDuplicates(array);

System.out.println("New length: " + newLength);

}

}

1. **Write a recursive Java program to calculate the factorial of a given number. • Input: A single integer. • Output: The factorial of the integer.**

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

long factorial = factorial(num);

System.out.println("Factorial: " + factorial);

}

public static long factorial(int n) {

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

return 1;

} else {

return n \* factorial(n - 1);

}

}

}

1. **Write a Java program to find the length of the longest common subsequence between two strings. • Input: Two strings. • Output: The length of the longest common subsequence.**

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the first string:");

String str1 = scanner.nextLine();

System.out.println("Enter the second string:");

String str2 = scanner.nextLine();

int length = longestCommonSubsequence(str1, str2);

System.out.println("Length of longest common subsequence: " + length);

}

public static int longestCommonSubsequence(String str1, String str2) {

int m = str1.length();

int n = str2.length();

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

return 0;

}

if (str1.charAt(m - 1) == str2.charAt(n - 1)) {

return 1 + longestCommonSubsequence(str1.substring(0, m - 1), str2.substring(0, n - 1));

} else {

return Math.max(longestCommonSubsequence(str1, str2.substring(0, n - 1)), longestCommonSubsequence(str1.substring(0, m - 1), str2));

}

}

}

1. **Write a Java program to reverse a given string without using any built-in string manipulation functions. • Input: A single string. • Output: The reversed string.**

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

String reversed = reverse(str);

System.out.println("Reversed string: " + reversed);

}

public static String reverse(String str) {

char[] chars = new char[str.length()];

int j = 0;

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

chars[j++] = str.charAt(i);

}

return new String(chars);

}

}

1. **Write a Java program to check if a given number is an Armstrong number. An Armstrong number for a given number of digits is a number whose sum of its own digits each raised to the power of the number of digits is equal to the number itself. • Input: A single integer. • Output: Print "Armstrong" if the number is an Armstrong number, otherwise print "Not Armstrong".**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

if (isArmstrong(num)) {

System.out.println("Armstrong");

} else {

System.out.println("Not Armstrong");

}

}

public static boolean isArmstrong(int num) {

int sum = 0;

int digits = String.valueOf(num).length();

for (int i = num; i > 0; i /= 10) {

int digit = i % 10;

sum += Math.pow(digit, digits);

}

return sum == num;

}

}

1. **Write a Java program to count the number of vowels and consonants in a given string. • Input: A single string. • Output: Print the count of vowels and consonants.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

int vowelCount = 0;

int consonantCount = 0;

for (char c : str.toLowerCase().toCharArray()) {

if (c >= 'a' && c <= 'z') {

if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {

vowelCount++;

} else {

consonantCount++;

}

}

}

System.out.println("Vowel count: " + vowelCount);

System.out.println("Consonant count: " + consonantCount);

}

}

1. **Write a Java program to find the greatest common divisor (GCD) of two given numbers using Euclid's algorithm. • Input: Two integers. • Output: The GCD of the two integers.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num1 = scanner.nextInt();

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

int num2 = scanner.nextInt();

int gcd = findGCD(num1, num2);

System.out.println("GCD: " + gcd);

}

public static int findGCD(int num1, int num2) {

if (num2 == 0) {

return num1;

} else {

return findGCD(num2, num1 % num2);

}

}

}

1. **Write a Java program to implement the bubble sort algorithm to sort an array of integers in ascending order. • Input: An unsorted array of integers. • Output: The sorted array**.

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int[] arr = new int[n];

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

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

arr[i] = scanner.nextInt();

}

bubbleSort(arr);

System.out.println("Sorted array:");

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

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

}

}

public static void bubbleSort(int[] arr) {

int n = arr.length;

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

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

if (arr[j] > arr[j + 1]) {

// Swap arr[j] and arr[j + 1]

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

}

1. **Write a Java program to check if a given string is a palindrome, ignoring spaces and case. • Input: A single string. • Output: Print "Palindrome" if the string is a palindrome, otherwise print "Not a Palindrome".**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

if (isPalindrome(str)) {

System.out.println("Palindrome");

} else {

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

}

}

public static boolean isPalindrome(String str) {

str = str.replaceAll("\\s+", "").toLowerCase();

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false;

}

left++;

right--;

}

return true;

}

}

1. **Write a Java program to find all prime numbers up to a given number using the Sieve of Eratosthenes. • Input: A single integer n. • Output: A list of prime numbers less than or equal to n.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

boolean[] sieve = new boolean[n + 1];

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

sieve[i] = true;

}

for (int i = 2; i \* i <= n; i++) {

if (sieve[i]) {

for (int j = i \* i; j <= n; j += i) {

sieve[j] = false;

}

}

}

System.out.println("Prime numbers:");

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

if (sieve[i]) {

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

}

}

}

}

1. **Write a Java program to remove duplicate characters from a given string. • Input: A single string. • Output: The string with duplicates removed**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

String result = removeDuplicates(str);

System.out.println("String with duplicates removed:");

System.out.println(result);

}

public static String removeDuplicates(String str) {

StringBuilder result = new StringBuilder();

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

if (result.indexOf(String.valueOf(c)) == -1) {

result.append(c);

}

}

return result.toString();

}

}

1. **Write a Java program to find the second largest element in an array of integers. • Input: An array of integers. • Output: The second largest element.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

int[] arr = new int[n];

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

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

arr[i] = scanner.nextInt();

}

int secondLargest = findSecondLargest(arr);

System.out.println("Second largest element: " + secondLargest);

}

public static int findSecondLargest(int[] arr) {

int max = Integer.MIN\_VALUE;

int secondMax = Integer.MIN\_VALUE;

for (int num : arr) {

if (num > max) {

secondMax = max;

max = num;

} else if (num > secondMax && num != max) {

secondMax = num;

}

}

return secondMax;

}

}

1. **Write a Java program to find the length of the longest substring without repeating characters in a given string. • Input: A single string. • Output: The length of the longest substring without repeating characters.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str = scanner.nextLine();

int length = findLongestSubstringLength(str);

System.out.println("Length of the longest substring without repeating characters: " + length);

}

public static int findLongestSubstringLength(String str) {

int maxLength = 0;

int left = 0;

boolean[] charSet = new boolean[256];

for (int right = 0; right < str.length(); right++) {

while (charSet[str.charAt(right)]) {

charSet[str.charAt(left)] = false;

left++;

}

charSet[str.charAt(right)] = true;

maxLength = Math.max(maxLength, right - left + 1);

}

return maxLength;

}

}

1. **Write a Java program to calculate x raised to the power y (x^y) without using built-in math functions. • Input: Two integers x and y. • Output: The result of x^y.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the base (x):");

int x = scanner.nextInt();

System.out.println("Enter the exponent (y):");

int y = scanner.nextInt();

int result = power(x, y);

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

}

public static int power(int x, int y) {

int result = 1;

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

result \*= x;

}

return result;

}

}

1. **Write a Java program to merge two sorted arrays into a single sorted array. • Input: Two sorted arrays of integers. • Output: A merged sorted array**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n1 = scanner.nextInt();

int[] arr1 = new int[n1];

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

for (int i = 0; i < n1; i++) {

arr1[i] = scanner.nextInt();

}

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

int n2 = scanner.nextInt();

int[] arr2 = new int[n2];

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

for (int i = 0; i < n2; i++) {

arr2[i] = scanner.nextInt();

}

int[] merged = mergeSortedArrays(arr1, arr2);

System.out.println("Merged sorted array:");

for (int num : merged) {

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

}

}

public static int[] mergeSortedArrays(int[] arr1, int[] arr2) {

int[] merged = new int[arr1.length + arr2.length];

int i = 0, j = 0, k = 0;

while (i < arr1.length && j < arr2.length) {

if (arr1[i] < arr2[j]) {

merged[k++] = arr1[i++];

} else {

merged[k++] = arr2[j++];

}

}

while (i < arr1.length) {

merged[k++] = arr1[i++];

}

while (j < arr2.length) {

merged[k++] = arr2[j++];

}

return merged;

}

}

1. **Write a Java program to check if two strings are rotations of each other. • Input: Two strings. • Output: Print "Rotation" if one string is a rotation of the other, otherwise print "Not Rotation".**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the first string:");

String str1 = scanner.nextLine();

System.out.println("Enter the second string:");

String str2 = scanner.nextLine();

if (isRotation(str1, str2)) {

System.out.println("Rotation");

} else {

System.out.println("Not Rotation");

}

}

public static boolean isRotation(String str1, String str2) {

if (str1.length() != str2.length()) {

return false;

}

String temp = str1 + str1;

return temp.contains(str2);

}

}

1. **Write a Java program to print the first n Fibonacci numbers using recursion. • Input: An integer n. • Output: The first n Fibonacci numbers**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of Fibonacci numbers to print:");

int n = scanner.nextInt();

System.out.println("First " + n + " Fibonacci numbers:");

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

System.out.print(fibonacci(i) + " ");

}

}

public static int fibonacci(int n) {

if (n <= 1) {

return n;

} else {

return fibonacci(n - 1) + fibonacci(n - 2);

}

}

}

1. **Write a Java program to check if a given number is a perfect number. A perfect number is a positive integer that is equal to the sum of its proper divisors, excluding itself. • Input: A single integer. • Output: Print "Perfect Number" if the number is perfect, otherwise print "Not a Perfect Number".**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int num = scanner.nextInt();

if (isPerfectNumber(num)) {

System.out.println("Perfect Number");

} else {

System.out.println("Not a Perfect Number");

}

}

public static boolean isPerfectNumber(int num) {

int sum = 0;

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

if (num % i == 0) {

sum += i;

}

}

return sum == num;

}

}

1. **Java program that converts a given decimal number into its binary and octal equivalents. The program also includes input validation to ensure the user enters a valid non-negative integer.**

import java.util.InputMismatchException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int decimal;

while (true) {

try {

System.out.println("Enter a non-negative integer:");

decimal = scanner.nextInt();

if (decimal < 0) {

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

} else {

break;

}

} catch (InputMismatchException e) {

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

scanner.next();

}

}

String binary = decimalToBinary(decimal);

String octal = decimalToOctal(decimal);

System.out.println("Binary equivalent: " + binary);

System.out.println("Octal equivalent: " + octal);

}

public static String decimalToBinary(int decimal) {

return Integer.toBinaryString(decimal);

}

public static String decimalToOctal(int decimal) {

return Integer.toOctalString(decimal);

}

}

1. **Write a program that takes the marks of five subjects and calculates the total, average, and grade of a student. The grade is determined as follows: A for average >= 90, B for average >= 80 and < 90, C for average >= 70 and < 80, D for average >= 60 and < 70, and F for average < 60.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int[] marks = new int[5];

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

System.out.println("Enter marks for subject " + (i + 1) + ":");

marks[i] = scanner.nextInt();

}

int total = calculateTotal(marks);

double average = calculateAverage(total, marks.length);

char grade = calculateGrade(average);

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

System.out.println("Average: " + average);

System.out.println("Grade: " + grade);

}

public static int calculateTotal(int[] marks) {

int total = 0;

for (int mark : marks) {

total += mark;

}

return total;

}

public static double calculateAverage(int total, int numSubjects) {

return (double) total / numSubjects;

}

public static char calculateGrade(double average) {

if (average >= 90) {

return 'A';

} else if (average >= 80) {

return 'B';

} else if (average >= 70) {

return 'C';

} else if (average >= 60) {

return 'D';

} else {

return 'F';

}

}

}

1. **Write a program to calculate the electricity bill based on the units consumed. The charges per unit are as follows: o First 100 units: $1.50 per unit o Next 200 units: $2.00 per unit o Above 300 units: $3.00 per unit • Additionally, if the total bill exceeds $500, a surcharge of 10% is added.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of units consumed:");

int units = scanner.nextInt();

double bill = calculateBill(units);

double surcharge = calculateSurcharge(bill);

double totalBill = bill + surcharge;

System.out.println("Total Bill: $" + totalBill);

}

public static double calculateBill(int units) {

double bill = 0;

if (units <= 100) {

bill = units \* 1.5;

} else if (units <= 300) {

bill = 100 \* 1.5 + (units - 100) \* 2;

} else {

bill = 100 \* 1.5 + 200 \* 2 + (units - 300) \* 3;

}

return bill;

}

public static double calculateSurcharge(double bill) {

if (bill > 500) {

return bill \* 0.1;

} else {

return 0;

}

}

}

1. **Write a program to manage a library system. The system should track books, which have a title, author, and availability status. It should allow users to borrow and return books, and it should ensure that books are not borrowed if they are already checked out.**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Book {

String title;

String author;

boolean isAvailable;

public Book(String title, String author) {

this.title = title;

this.author = author;

this.isAvailable = true;

}

}

public class LibrarySystem {

Map<String, Book> books = new HashMap<>();

Scanner scanner = new Scanner(System.in);

public void addBook() {

System.out.println("Enter book title:");

String title = scanner.nextLine();

System.out.println("Enter book author:");

String author = scanner.nextLine();

books.put(title, new Book(title, author));

System.out.println("Book added successfully!");

}

public void borrowBook() {

System.out.println("Enter book title:");

String title = scanner.nextLine();

if (books.containsKey(title)) {

Book book = books.get(title);

if (book.isAvailable) {

book.isAvailable = false;

System.out.println("Book borrowed successfully!");

} else {

System.out.println("Book is already checked out!");

}

} else {

System.out.println("Book not found!");

}

}

public void returnBook() {

System.out.println("Enter book title:");

String title = scanner.nextLine();

if (books.containsKey(title)) {

Book book = books.get(title);

if (!book.isAvailable) {

book.isAvailable = true;

System.out.println("Book returned successfully!");

} else {

System.out.println("Book is already available!");

}

} else {

System.out.println("Book not found!");

}

}

public void displayBooks() {

for (Book book : books.values()) {

System.out.println("Title: " + book.title + ", Author: " + book.author + ", Availability: " + (book.isAvailable ? "Yes" : "No"));

}

}

public static void main(String[] args) {

LibrarySystem librarySystem = new LibrarySystem();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Add Book");

System.out.println("2. Borrow Book");

System.out.println("3. Return Book");

System.out.println("4. Display Books");

System.out.println("5. Exit");

System.out.println("Enter your choice:");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

switch (choice) {

case 1:

librarySystem.addBook();

break;

case 2:

librarySystem.borrowBook();

break;

case 3:

librarySystem.returnBook();

break;

case 4:

librarySystem.displayBooks();

break;

case 5:

System.exit(0);

break;

default:

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

}

}

}

}

1. **Write a program to calculate income tax based on the following slab: o Income up to $10,000: No tax o Income from $10,001 to $20,000: 10% tax o Income from $20,001 to $50,000: 20% tax o Income above $50,000: 30% tax • The program should take the user’s income as input and calculate the tax accordingly.**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter your income:");

double income = scanner.nextDouble();

double tax = calculateTax(income);

System.out.println("Your income tax is: $" + tax);

}

public static double calculateTax(double income) {

if (income <= 10000) {

return 0;

} else if (income <= 20000) {

return (income - 10000) \* 0.10;

} else if (income <= 50000) {

return (10000 \* 0.10) + ((income - 20000) \* 0.20);

} else {

return (10000 \* 0.10) + (30000 \* 0.20) + ((income - 50000) \* 0.30);

}

}

}

1. **Create a simple ATM system where users can check their balance, deposit money, and withdraw money. The system should ensure that withdrawals do not exceed the available balance and should update the balance after each transaction.**

import java.util.Scanner;

public class ATM {

private double balance;

public ATM(double balance) {

this.balance = balance;

}

public void checkBalance() {

System.out.println("Your current balance is: $" + balance);

}

public void deposit(double amount) {

balance += amount;

System.out.println("Deposit successful. New balance is: $" + balance);

}

public void withdraw(double amount) {

if (amount > balance) {

System.out.println("Insufficient funds!");

} else {

balance -= amount;

System.out.println("Withdrawal successful. New balance is: $" + balance);

}

}

public static void main(String[] args) {

ATM atm = new ATM(1000); // Initial balance is $1000

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Check Balance");

System.out.println("2. Deposit");

System.out.println("3. Withdraw");

System.out.println("4. Exit");

System.out.println("Enter your choice:");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

atm.checkBalance();

break;

case 2:

System.out.println("Enter amount to deposit:");

double depositAmount = scanner.nextDouble();

scanner.nextLine();

atm.deposit(depositAmount);

break;

case 3:

System.out.println("Enter amount to withdraw:");

double withdrawAmount = scanner.nextDouble();

scanner.nextLine();

atm.withdraw(withdrawAmount);

break;

case 4:

System.exit(0);

break;

default:

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

}

}

}

}

1. **Develop a payroll system that calculates the net salary of employees. The salary should include base pay and various allowances (house rent allowance, dearness allowance) and deductions (tax, provident fund). The allowances are a percentage of the base pay, and the deductions are either fixed or percentage-based**

import java.util.Scanner;

public class PayrollSystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter employee name:");

String name = scanner.nextLine();

System.out.println("Enter base pay:");

double basePay = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.println("Enter house rent allowance percentage:");

double hraPercentage = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.println("Enter dearness allowance percentage:");

double daPercentage = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.println("Enter tax percentage:");

double taxPercentage = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.println("Enter provident fund percentage:");

double pfPercentage = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

double hra = (hraPercentage / 100) \* basePay;

double da = (daPercentage / 100) \* basePay;

double tax = (taxPercentage / 100) \* basePay;

double pf = (pfPercentage / 100) \* basePay;

double grossSalary = basePay + hra + da;

double netSalary = grossSalary - tax - pf;

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

System.out.println("Gross Salary: $" + grossSalary);

System.out.println("Net Salary: $" + netSalary);

}

}

1. **Implement an online shopping cart where users can add and remove items, view the total price of items in the cart, and apply discounts if applicable (e.g., a discount code). The program should handle different product prices and quantities.**

import java.util.Scanner;

public class ShoppingCart {

double total = 0;

Scanner scanner = new Scanner(System.in);

public void addProduct(double price, int quantity) {

total += price \* quantity;

}

public void removeProduct(double price, int quantity) {

total -= price \* quantity;

}

public void viewTotal() {

System.out.println("Total: $" + total);

}

public void applyDiscount(double discountPercentage) {

double discountAmount = (discountPercentage / 100) \* total;

System.out.println("Discount applied: $" + discountAmount);

System.out.println("New total: $" + (total - discountAmount));

}

public static void main(String[] args) {

ShoppingCart cart = new ShoppingCart();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Add product");

System.out.println("2. Remove product");

System.out.println("3. View total");

System.out.println("4. Apply discount");

System.out.println("5. Exit");

System.out.println("Enter your choice:");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

switch (choice) {

case 1:

System.out.println("Enter product price:");

double price = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.println("Enter quantity:");

int quantity = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

cart.addProduct(price, quantity);

break;

case 2:

System.out.println("Enter product price:");

price = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

System.out.println("Enter quantity:");

quantity = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

cart.removeProduct(price, quantity);

break;

case 3:

cart.viewTotal();

break;

case 4:

System.out.println("Enter discount percentage:");

double discountPercentage = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

cart.applyDiscount(discountPercentage);

break;

case 5:

System.exit(0);

break;

default:

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

}

}

}

}

1. **Write a Java program that safely handles the division of two numbers and specifically checks for division by zero. If division by zero is attempted, the program should catch the exception and provide an appropriate error message.**

import java.util.InputMismatchException;

import java.util.Scanner;

public class SafeDivision {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (true) {

try {

System.out.println("Enter dividend:");

double dividend = scanner.nextDouble();

System.out.println("Enter divisor:");

double divisor = scanner.nextDouble();

if (divisor == 0) {

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

}

double result = dividend / divisor;

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

} catch (InputMismatchException e) {

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

scanner.next(); // Clear invalid input

} catch (ArithmeticException e) {

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

}

}

}

}

1. **Write a Java program that takes an array of integers from the user and an index position. The program should attempt to access the element at the given index and divide it by another number provided by the user. Handle the following exceptions: • ArrayIndexOutOfBoundsException: If the index is out of the array bounds. • ArithmeticException: If there is an attempt to divide by zero. • Any other general exception.**

import java.util.Scanner;

public class ArrayDivision {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int size = scanner.nextInt();

int[] array = new int[size];

System.out.println("Enter " + size + " elements:");

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

array[i] = scanner.nextInt();

}

System.out.println("Enter the index position:");

int index = scanner.nextInt();

System.out.println("Enter the divisor:");

int divisor = scanner.nextInt();

try {

int result = array[index] / divisor;

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

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Index out of bounds");

} catch (ArithmeticException e) {

System.out.println("Error: Division by zero");

} catch (Exception e) {

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

}

}

}

1. **Write a Java program to simulate an age validation system. The program should throw a custom InvalidAgeException if the user enters an age less than 18. Use throw to explicitly throw the exception and throws to declare it.**

class InvalidAgeException extends Exception {

public InvalidAgeException(String message) {

super(message);

}

}

public class AgeValidation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter your age:");

int age = scanner.nextInt();

try {

if (age < 18) {

throw new InvalidAgeException("Age must be 18 or older");

} else {

System.out.println("Age is valid");

}

} catch (InvalidAgeException e) {

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

}

}

}

1. **Write a Java program that simulates a bank account. The program should throw a custom InsufficientFundsException if a withdrawal amount exceeds the account balance. The program should also throw a custom NegativeAmountException if the withdrawal or deposit amount is negative.**

class InsufficientFundsException extends Exception {

public InsufficientFundsException(String message) {

super(message);

}

}

class NegativeAmountException extends Exception {

public NegativeAmountException(String message) {

super(message);

}

}

public class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

public void deposit(double amount) throws NegativeAmountException {

if (amount < 0) {

throw new NegativeAmountException("Deposit amount cannot be negative");

}

balance += amount;

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

}

public void withdraw(double amount) throws InsufficientFundsException, NegativeAmountException {

if (amount < 0) {

throw new NegativeAmountException("Withdrawal amount cannot be negative");

}

if (amount > balance) {

throw new InsufficientFundsException("Insufficient funds for withdrawal");

}

balance -= amount;

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

}

public static void main(String[] args) {

BankAccount account = new BankAccount(1000);

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Exit");

System.out.println("Enter your choice:");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.println("Enter deposit amount:");

double depositAmount = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

try {

account.deposit(depositAmount);

} catch (NegativeAmountException e) {

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

}

break;

case 2:

System.out.println("Enter withdrawal amount:");

double withdrawalAmount = scanner.nextDouble();

scanner.nextLine(); // Consume newline left-over

try {

account.withdraw(withdrawalAmount);

} catch (InsufficientFundsException | NegativeAmountException e) {

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

}

break;

case 3:

System.exit(0);

break;

default:

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

}

}

}

}

1. **Write a Java program that demonstrates nested try-catch blocks, where the outer block handles a general exception and the inner block handles specific exceptions like ArithmeticException and ArrayIndexOutOfBoundsException**.

public class NestedTryCatch {

public static void main(String[] args) {

try {

try {

int[] array = new int[5];

array[10] = 10;

int dividend = 10;

int divisor = 0;

int result = dividend / divisor;

} catch (ArithmeticException e) {

System.out.println("Inner catch: ArithmeticException - " + e.getMessage());

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Inner catch: ArrayIndexOutOfBoundsException - " + e.getMessage());

}

} catch (Exception e) {

System.out.println("Outer catch: General Exception - " + e.getMessage());

}

}

}

1. **Write a Java program that reads from a file using try-withresources, ensuring that the file resource is closed automatically even if an exception occurs.**

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class ReadFile {

public static void main(String[] args) {

File file = new File("example.txt");

try (Scanner scanner = new Scanner(file)) {

while (scanner.hasNextLine()) {

System.out.println(scanner.nextLine());

}

} catch (FileNotFoundException e) {

System.out.println("File not found: " + e.getMessage());

}

}

}

1. **Write a Java program that demonstrates exception chaining, where a custom exception wraps another exception.**

class CustomException extends Exception {

public CustomException(String message, Throwable cause) {

super(message, cause);

}

}

public class ExceptionChaining {

public static void main(String[] args) {

try {

try {

throw new ArithmeticException("Division by zero");

} catch (ArithmeticException e) {

throw new CustomException("Custom exception wrapping ArithmeticException", e);

}

} catch (CustomException e) {

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

Throwable cause = e.getCause();

while (cause != null) {

System.out.println("Caused by: " + cause.getMessage());

cause = cause.getCause();

}

}

}

}

1. **Write a Java program to demonstrate the use of a finally block, which executes regardless of whether an exception is thrown or not.**

public class FinallyBlock {

public static void main(String[] args) {

try {

System.out.println("Try block");

int result = 10 / 0; // ArithmeticException

} catch (ArithmeticException e) {

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

} finally {

System.out.println("Finally block");

}

System.out.println("Program continues...");

}

}

1. **Write a Java program that catches an exception, processes it, and then rethrows it to be handled by another catch block or by the calling method.**

public class RethrowException {

public static void main(String[] args) {

try {

try {

// Simulate an exception

throw new ArithmeticException("Division by zero");

} catch (ArithmeticException e) {

// Process the exception

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

// Rethrow the exception

throw e;

}

} catch (ArithmeticException e) {

// Handle the rethrown exception

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

}

}

}

1. **Write a Java program that prints the first n perfect numbers. A perfect number is a positive integer that is equal to the sum of its proper divisors (excluding itself).**

import java.util.Scanner;

public class PerfectNumbers {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

System.out.println(“Enter the number of perfect numbers :”);

int n =input.nextInt();

int count = 0;

int num = 2;

while (count < n) {

if (isPerfect(num)) {

System.out.println(num);

count++;

}

num++;

}

}

public static boolean isPerfect(int num) {

int sum = 0;

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

if (num % i == 0) {

sum += i;

}

}

return sum == num;

}

}

1. **Write a Java program to find the largest and smallest elements in an array of integers.**

public class LargestAndSmallest {

public static void main(String[] args) {

int[] array = {12, 45, 7, 23, 56, 89, 34};

int largest = findLargest(array);

int smallest = findSmallest(array);

System.out.println("Largest element: " + largest);

System.out.println("Smallest element: " + smallest);

}

public static int findLargest(int[] array) {

int largest = array[0];

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

if (array[i] > largest) {

largest = array[i];

}

}

return largest;

}

public static int findSmallest(int[] array) {

int smallest = array[0];

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

if (array[i] < smallest) {

smallest = array[i];

}

}

return smallest;

}

}

1. **Write a Java program to reverse a given string**

import java.util.Scanner;

public class ReverseString {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String input = scanner.nextLine();

String reversed = reverse(input);

System.out.println("Reversed string: " + reversed);

}

public static String reverse(String str) {

String reversed = "";

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

reversed += str.charAt(i);

}

return reversed;

}

}

1. **Write a Java program to calculate and print the Fibonacci series up to n terms**.

import java.util.Scanner;

public class Fibonacci {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of terms:");

int n = scanner.nextInt();

printFibonacci(n);

}

public static void printFibonacci(int n) {

int t1 = 0, t2 = 1;

System.out.println("Fibonacci series up to " + n + " terms:");

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

System.out.print(t1 + " ");

int sum = t1 + t2;

t1 = t2;

t2 = sum;

}

}

}

1. **Write a Java program to check if a given string is a palindrome.**

import java.util.Scanner;

public class Palindrome {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String input = scanner.nextLine();

if (isPalindrome(input)) {

System.out.println(input + " is a palindrome.");

} else {

System.out.println(input + " is not a palindrome.");

}

}

public static boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false;

}

left++;

right--;

}

return true;

}

}

1. **Write a Java program to sort an array using the bubble sort algorithm.**

public class BubbleSort {

public static void main(String[] args) {

int[] array = {64, 34, 25, 12, 22, 11, 90};

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

printArray(array);

bubbleSort(array);

System.out.println("Sorted array:");

printArray(array);

}

public static void bubbleSort(int[] array) {

int n = array.length;

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

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

if (array[j] > array[j + 1]) {

// Swap array[j] and array[j+1]

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

public static void printArray(int[] array) {

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

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

}

System.out.println();

}

}

1. **Write a Java program to merge two sorted arrays into a single sorted array.**

public class MergeSortedArrays {

public static void main(String[] args) {

int[] array1 = {1, 3, 5, 7};

int[] array2 = {2, 4, 6, 8};

int[] mergedArray = mergeSortedArrays(array1, array2);

System.out.println("Merged sorted array:");

printArray(mergedArray);

}

public static int[] mergeSortedArrays(int[] array1, int[] array2) {

int[] mergedArray = new int[array1.length + array2.length];

int i = 0, j = 0, k = 0;

while (i < array1.length && j < array2.length) {

if (array1[i] < array2[j]) {

mergedArray[k++] = array1[i++];

} else {

mergedArray[k++] = array2[j++];

}

}

while (i < array1.length) {

mergedArray[k++] = array1[i++];

}

while (j < array2.length) {

mergedArray[k++] = array2[j++];

}

return mergedArray;

}

public static void printArray(int[] array) {

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

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

}

System.out.println();

}

}

1. **Write a Java program to sort an array using the Quick Sort algorithm.**

public class QuickSort {

public static void main(String[] args) {

int[] array = {5, 2, 9, 1, 7, 3};

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

printArray(array);

quickSort(array, 0, array.length - 1);

System.out.println("Sorted array:");

printArray(array);

}

public static void quickSort(int[] array, int low, int high) {

if (low < high) {

int pivotIndex = partition(array, low, high);

quickSort(array, low, pivotIndex - 1);

quickSort(array, pivotIndex + 1, high);

}

}

public static int partition(int[] array, int low, int high) {

int pivot = array[high];

int i = low - 1;

for (int j = low; j < high; j++) {

if (array[j] < pivot) {

i++;

swap(array, i, j);

}

}

swap(array, i + 1, high);

return i + 1;

}

public static void swap(int[] array, int i, int j) {

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

public static void printArray(int[] array) {

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

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

}

System.out.println();

}

}

1. **Write a Java program to find all prime numbers up to a given number n using the Sieve of Eratosthenes algorithm.**

import java.util.Scanner;

public class SieveOfEratosthenes {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a number n:");

int n = scanner.nextInt();

boolean[] prime = new boolean[n + 1];

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

prime[i] = true;

}

prime[0] = prime[1] = false;

for (int p = 2; p \* p <= n; p++) {

// If p is prime, mark as composite all the multiples of p

if (prime[p]) {

for (int i = p \* p; i <= n; i += p) {

prime[i] = false;

}

}

}

System.out.println("Prime numbers up to " + n + ":");

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

if (prime[i]) {

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

}

}

}

}

1. **Write a Java program to find the longest common prefix string amongst an array of strings**.

public class LongestCommonPrefix {

public static void main(String[] args) {

String[] strs = {"flower", "flow", "flight"};

String prefix = longestCommonPrefix(strs);

System.out.println("Longest common prefix: " + prefix);

}

public static String longestCommonPrefix(String[] strs) {

if (strs.length == 0) {

return "";

}

String prefix = strs[0];

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

while (strs[i].indexOf(prefix) != 0) {

prefix = prefix.substring(0, prefix.length() - 1);

if (prefix.isEmpty()) {

return "";

}

}

}

return prefix;

}

}

1. **Write a Java program to find the Kth largest element in an unsorted array.**

import java.util.Arrays;

public class KthLargestElement {

public static void main(String[] args) {

int[] array = {12, 3, 5, 7, 19, 4, 1};

int k = 3;

int kthLargest = findKthLargest(array, k);

System.out.println("Kth largest element: " + kthLargest);

}

public static int findKthLargest(int[] array, int k) {

Arrays.sort(array);

return array[array.length - k];

}

}

1. **Write a Java program to create and start two threads. Each thread should print a message indicating which thread it is, and the current thread ID. Use both Thread class and Runnable interface approaches.**

class ThreadExample extends Thread {

public void run() {

System.out.println("Thread ID: " + Thread.currentThread().getId() + " - This is ThreadExample thread");

}

}

class RunnableExample implements Runnable {

public void run() {

System.out.println("Thread ID: " + Thread.currentThread().getId() + " - This is RunnableExample thread");

}

}

public class Main {

public static void main(String[] args) {

// Using Thread class

ThreadExample thread1 = new ThreadExample();

thread1.start();

// Using Runnable interface

RunnableExample runnable = new RunnableExample();

Thread thread2 = new Thread(runnable);

thread2.start();

}

}

1. **Write a Java program with a class Counter that has a synchronized method increment() to increment a count. Create two threads that each increment the count 1000 times. Print the final count value.**

class Counter {

private int count = 0;

public synchronized void increment() {

count++;

}

public int getCount() {

return count;

}

}

class CounterThread extends Thread {

private Counter counter;

public CounterThread(Counter counter) {

this.counter = counter;

}

public void run() {

for (int i = 0; i < 1000; i++) {

counter.increment();

}

}

}

public class Main {

public static void main(String[] args) throws InterruptedException {

Counter counter = new Counter();

CounterThread thread1 = new CounterThread(counter);

CounterThread thread2 = new CounterThread(counter);

thread1.start();

thread2.start();

thread1.join();

thread2.join();

System.out.println("Final count: " + counter.getCount());

}

}

1. **Write a Java program to use ScheduledExecutorService to schedule a task that prints the current time every 2 seconds. Schedule another task to shut down the scheduler after 10 seconds.**

import java.time.LocalTime;

import java.util.concurrent.Executors;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.TimeUnit;

public class SchedulerExample {

public static void main(String[] args) {

ScheduledExecutorService scheduler = Executors.newScheduledThreadPool(2);

// Schedule a task to print the current time every 2 seconds

scheduler.scheduleAtFixedRate(() -> {

System.out.println("Current time: " + LocalTime.now());

}, 0, 2, TimeUnit.SECONDS);

// Schedule another task to shut down the scheduler after 10 seconds

scheduler.schedule(() -> {

System.out.println("Shutting down the scheduler");

scheduler.shutdown();

}, 10, TimeUnit.SECONDS);

}

}