**1.1. Accept two strings from the user and print it on console with concatenation of “and” in middle of**

**the strings.**

**Class Args**

**{**

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

**{**

**System.out.println(args[0]+”and ”+args[1]);**

**}**

**}**

**The above program save as Args.java**

**Compilation of the program is javac Args.java**

**Execution of the Program is**

**Java Args C Java**

**Output: C and Java**

**1.2. Accept 12-hour time zone and convert into it’s corresponding 24-hour time zone.**

**Class TimeFormat**

**{**

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

**{**

**Int a,b,c;**

**System.out.println(“Enter the time”);**

**a=Integer.parseInt(args[0]);**

**b=Integer.parseInt(args[1]);**

**c=Integer.parseInt(args[2]);**

**if(args[3].equls(”am”))**

**{**

**Enter the time**

**System.out.println(a+”:”+b+”:”+c);**

**}**

**else**

**{**

**a=a+12;**

**System.out.println(a+”:”+b+”:”+c);**

**}**

**}**

**}**

**The above program save as TimeFormat.java**

**Compilation of the program is javac TimeFormat.java**

**Execution of the Program is**

**Java TimeFormat**

**Output: .**

**Enter the time**

**07 05 45 PM**

**19:05:45 PM**

**1.3 Accept three subject marks of a student and calculate the percentage and determine the division of**

**the student. (e.g. &gt;=70 distinction, &gt;60 and &lt;70 first class…so on). Print percentage in both**

**float and double formats.**

**Class Percent{**

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

**{**

**Int m1,m2,m3;**

**m1= Integer.parseInt(args[0]);**

**m2= Integer.parseInt(args[1]);**

**m3= Integer.parseInt(args[3]);**

**float percentage=(m1+m2+m3)/3;**

**double percent=(m1+m2+m3)/3;**

**if (percentage&gt;70.0)**

**System.out.println(“Destinction”);**

**else if (percentage&gt;60.0) and (percentage&lt;70.0)**

**System.out.println(“ First class”);**

**else if (percentage&gt;50.0) and (percentage&lt;60.0)**

**System.out.println(“ Second class”);**

**else if (percentage&gt;40.0)**

**System.out.println(“ PASS”);**

**else**

**{**

**System.out.println(“ Failed”);**

**}**

**System.out.println(“float percentage :” +percentage);**

**System.out.println(“double percentage :” +percent);**

**}**

**}**

**1.B perform the above program using Scanner class**

**Accept two strings from the user and print it on console with concatenation of “and” in middle of the**

**strings.**

**Imprt java.util.\*;**

**Class Args1**

**{**

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

**{**

**Scanner sc=new scanner(System.in);**

**System.out.println(“Enter two Strings :”);**

**String s1=sc.nextLine();**

**String s2=sc.nextLine();**

**System.out.println(s1+ “ and ” +s2);**

**}**

**}**

**1.2. Accept 12-hour time zone and convert into it’s corresponding 24-hour time zone.**

**Import java.util.\*;**

**Class TmFt**

**{**

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

**{**

**Scanner sc=new scanner(System.in);**

**System.out.println(“Enter the time”);**

**Int a=sc.nextInt();**

**Int b= sc.nextInt();**

**Int c= sc.nextInt();**

**String d= sc.nextLine();**

**if(d.equls(”am”) || d.equls(”AM”))**

**{**

**System.out.println(a+”:”+b+”:”+c);**

**}**

**else**

**{**

**a=a+12;**

**System.out.println(a+”:”+b+”:”+c);**

**}**

**}**

**}**

**The above program save as TmFt.java**

**Compilation of the program is javac TmFt.java**

**Execution of the Program is**

**Java TmFt**

**Output: .**

**Enter the time**

**07 05 45 PM**

**19:05:45 PM**

**1.3 Accept three subject marks of a student and calculate the percentage and determine the division of**

**the student. (e.g. &gt;=70 distinction, &gt;60 and &lt;70 first class…so on). Print percentage in both**

**float and double formats.**

**Import java.util.\*;**

**Class Percent1{**

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

**{**

**Scanner sc=new scanner (System.in);**

**System.out.println(“Enter Marks for Three Sublects : ”);**

**Int m1,m2,m3;**

**m1= sc.nextInt();**

**m2= sc.nextInt();**

**m3= sc.nextInt();**

**float percentage=(m1+m2+m3)/3;**

**double percent=(m1+m2+m3)/3;**

**if (percentage&gt;70.0)**

**System.out.println(“Destinction”);**

**else if (percentage&gt;60.0 &amp;&amp; percentage&lt;70.0)**

**System.out.println(“ First class”);**

**else if (percentage&gt;50.0 &amp;&amp; percentage&lt;60.0)**

**System.out.println(“ Second class”);**

**else if (percentage&gt;40.0)**

**System.out.println(“ PASS”);**

**else**

**{**

**System.out.println(“ Failed”);**

**}**

**System.out.println(“float percentage :” +percentage);**

**System.out.println(“double percentage :” +percent);**

**}**

**}**

**1.4 Accept a number ‘n’ and print the list of ‘n’ Fibonacci terms recursively.**

**Import java.util.\*;**

**Class fib1{**

**{**

**Void val(int a,int b,int n)**

**{**

**If(n==0)**

**return;**

**else**

**{**

**System.out.println(a);**

**Int c=a+b;**

**a=b;**

**b=c;**

**n=n-1;**

**val(a,b,n);**

**}**

**}**

**Class fibonacci1**

**{**

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

**{**

**Scanner sc=new scanner (System.in);**

**System.out.println(“Enter n value : ”);**

**int n= sc.nextInt();**

**int a=0,b=1,**

**fib1 sc=new fib1();**

**sc.val(a,b,n);**

**}**

**}**

**2. Write a program that accepts set of inputs from the user of various integer data types and**

**determines the primitive data type that is capable of properly storing that input.**

**Import java.util.\*;**

**Class Primitive**

**{**

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

**{**

**int n,s;**

**short n1;**

**byte n2;**

**long n3;**

**double n4;**

**Scanner sc=new scanner (System.in);**

**s=sc.nextInt();**

**n=s;**

**n1=(short) s;**

**n2=(byte)s;**

**n3=(long)s;**

**n4=(double)s;**

**System.out.println(“Enter n value : ”+n);**

**System.out.println(“Enter n1 value : ”+n1);**

**System.out.println(“Enter n2 value : ”+n2);**

**System.out.println(“Enter n3 value : ”+n3);**

**System.out.println(“Enter n4 value : ”+n4);**

**}**

**}**

**2. B) Write a program that accepts an array of integers and print those which are both odd and prime.**

**If no such element in that array print “Not found”.**

**import java.util.\*;**

**class OddPrime {**

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

**Scanner sc = new Scanner(System.in);**

**System.out.println(&quot;Enter The Size of the Array: &quot;);**

**int n = sc.nextInt(); // Corrected &#39;Int&#39; to &#39;int&#39;**

**int a[] = new int[n]; // Declaring an array of size n**

**int flag, count = 0;**

**System.out.println(&quot;Enter the elements of the array:&quot;);**

**for (int i = 0; i &lt; n; i++) { // Fixed loop syntax**

**a[i] = sc.nextInt();**

**}**

**for (int i = 0; i &lt; n; i++) { // Looping through the array using index**

**if (a[i] % 2 != 0) { // Checking if the number is odd**

**flag = 1; // Assume the number is prime initially**

**for (int j = 2; j &lt; a[i]; j++) { // Checking for prime property**

**if (a[i] % j == 0) {**

**flag = 0; // Not prime**

**break;**

**}**

**}**

**if (flag == 1 &amp;&amp; a[i] &gt; 1) { // Ensuring it&#39;s prime and greater than 1**

**System.out.println(a[i] + &quot; is Both Odd and Prime.&quot;);**

**count++;**

**}**

**}**

**}**

**if (count == 0) {**

**System.out.println(&quot;Element Not Found&quot;);**

**}**

**sc.close(); // Close the scanner**

**}**

**}**

**2.c There are two players ‘A’ and ‘B’ plays a game ‘n’ times. The points gained by each player are**

**tabulated in two different arrays. Find the number of times each won the game.**

**import java.util.Scanner;**

**class PlayerGame {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for number of games**

**System.out.print(&quot;Enter the number of games: &quot;);**

**int n = sc.nextInt();**

**int player1[] = new int[n];**

**int player2[] = new int[n];**

**int player1\_wins = 0, player2\_wins = 0, draws = 0;**

**// Taking input for player 1 scores**

**System.out.println(&quot;Enter points of player A:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**player1[i] = sc.nextInt();**

**}**

**// Taking input for player 2 scores**

**System.out.println(&quot;Enter points of player B:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**player2[i] = sc.nextInt();**

**}**

**// Determining the winner for each game**

**for (int i = 0; i &lt; n; i++) {**

**if (player1[i] &gt; player2[i]) {**

**player1\_wins++;**

**} else if (player2[i] &gt; player1[i]) {**

**player2\_wins++;**

**} else {**

**draws++;**

**}**

**}**

**// Printing results**

**System.out.println(&quot;Player A won &quot; + player1\_wins + &quot; times.&quot;);**

**System.out.println(&quot;Player B won &quot; + player2\_wins + &quot; times.&quot;);**

**System.out.println(&quot;Number of draws: &quot; + draws);**

**sc.close();**

**}**

**}**

**Explanation**

**1. The program starts by asking the user for the number of games played (n).**

**2. Two integer arrays (player1 and player2) are created to store the scores of players A**

**and B.**

**3. The program takes input for both players&#39; scores.**

**4. A loop iterates through the scores and compares them to determine:**

**o If Player A wins, player1\_wins is incremented.**

**o If Player B wins, player2\_wins is incremented.**

**o If both players have the same score, the draws counter is incremented.**

**5. Finally, the program displays how many games each player won and the number of**

**draws.**

**Sample Input/Output**

**Input**

**Enter the number of games: 5**

**Enter points of player A:**

**10 20 30 40 50**

**Enter points of player B:**

**15 20 25 40 60**

**Output**

**Player A won 1 times.**

**Player B won 2 times.**

**Number of draws: 2**

**In this example:**

** Player A won once (game 3: 30 &gt; 25).**

** Player B won twice (game 1: 15 &gt; 10, game 5: 60 &gt; 50).**

** Two games were draws (game 2 and game 4: 20 == 20 and 40 == 40).**

**2.D. Write a program to accept contents into an Integer Array and print the frequency of each number**

**in the order of their number of occurrences.**

**import java.util.Scanner;**

**class Frequency {**

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

**Scanner sc = new Scanner (System.in);**

**// Taking input for array size**

**System.out.print(&quot;Enter the size of the array: &quot;);**

**int n = sc.nextInt();**

**int a[] = new int[n];**

**int count=0,I,j,p;**

**// Taking input for array elements**

**for (int i = 0; i &lt; n; i++) {**

**System.out.println(&quot;Enter element:&quot;);**

**a[i] = sc.nextInt();**

**}**

**for (int i = 0; i &lt; n; i++)**

**{**

**count=0;**

**p=a[i];**

**if(p!=0)**

**{**

**for (j = 0; j &lt; n; j++)**

**if(p==a[j])**

**{**

**Count++;**

**a[j]=0;**

**}**

**}**

**}**

**// Printing the frequencies**

**System.out.println(p+ &quot; --&gt; :&quot; +count);**

**}**

**}**

**}**

**Output:**

**Enter The size of the Array:6**

**Enter element:1**

**Enter element:2**

**Enter element:2**

**Enter element: 1**

**Enter element: 4**

**Enter element: 5**

**12**

**22**

**41**

**51**

**3. A) Write a program that accepts an ‘n’ ordered square matrix elements into a single dimension array**

**and print the elements of leading diagonal (top left to bottom right)**

**import java.util.Scanner;**

**class LeadingDiagonal {**

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

**Scanner sc = new Scanner(System.in);**

**// Input for matrix size (n x n)**

**System.out.print(&quot;Enter the order of the square matrix (n): &quot;);**

**int n = sc.nextInt();**

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

**int array[] = new int[n \* n];**

**// Taking matrix input and storing it in a 1D array**

**System.out.println(&quot;Enter the elements of the matrix:&quot;);**

**int index = 0;**

**for (int i = 0; i &lt; n; i++) {**

**for (int j = 0; j &lt; n; j++) {**

**matrix[i][j] = sc.nextInt();**

**array[index++] = matrix[i][j]; // Storing elements in 1D array**

**}**

**}**

**// Printing the matrix (for reference)**

**System.out.println(&quot;The Matrix is:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**for (int j = 0; j &lt; n; j++) {**

**System.out.print(matrix[i][j] + &quot; &quot;);**

**}**

**System.out.println();**

**}**

**// Printing the leading diagonal elements**

**System.out.println(&quot;Leading Diagonal Elements:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**System.out.print(matrix[i][i] + &quot; &quot;);**

**}**

**sc.close();**

**}**

**}**

**(OR)**

**import java.util.Scanner;**

**class Diagnol {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for array size**

**System.out.print(&quot;Enter the size of the array: &quot;);**

**int n = sc.nextInt();**

**int a[] = new int[n \* n];**

**// Accepting matrix elements**

**for (int i = 0; i &lt; n \* n; i++) {**

**System.out.println(&quot;Enter element:&quot;);**

**a[i] = sc.nextInt();**

**}**

**// Printing the leading diagonal elements**

**System.out.println(&quot;The Leading Diagonal of Matrix is:&quot;);**

**for (int j = 0; j &lt; n \* n; j = j + (n + 1)) {**

**System.out.println(a[j]);**

**}**

**sc.close();**

**}**

**}**

**Explanation of the Code**

**This Java program takes a square matrix as input, stores it in a 1D array, and then prints its**

**leading diagonal elements.**

**Step-by-Step Breakdown**

**1. User Input for Matrix Size:**

**o The user enters the size n of an n × n matrix.**

**o Since a matrix has n × n elements, an array of size n \* n is created.**

**2. Reading Matrix Elements:**

**o The program stores matrix elements in a 1D array instead of a 2D array.**

**o The user enters n × n elements one by one.**

**3. Extracting the Leading Diagonal:**

**o The leading diagonal consists of elements where row index = column index.**

**o In a 1D representation of an n × n matrix:**

** Formula for leading diagonal elements:**

** They occur at indices: 0, (n+1), 2(n+1), ... up to (n-**

**1)(n+1).**

** Using a loop, we traverse these indices with j = j + (n + 1).**

**4. Closing Scanner:**

**o sc.close(); is used to prevent memory leaks.**

**Output:**

**Enter the size of the Array: 3**

**Enter Element: 1**

**Enter Element: 7**

**Enter Element: 9**

**Enter Element: 2**

**Enter Element: 3**

**Enter Element: 5**

**Enter Element: 6**

**Enter Element: 9**

**Enter Element: 11**

**The Leading Diagnol of Matrix is**

**1**

**5**

**11**

**3.B) Write a program that accepts an ‘m x n’ double dimension array,**

**where ‘m’ represents financial years and ‘n’ represents Ids of the items**

**sold. Each element in the array represents number of items sold in a**

**particular year. Identify the year and id of the item which has more**

**demand**

**import java.util.Scanner;**

**class HighestDemand {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for number of years (m) and number of items (n)**

**System.out.print(&quot;Enter the number of financial years (m): &quot;);**

**int m = sc.nextInt();**

**System.out.print(&quot;Enter the number of items sold (n): &quot;);**

**int n = sc.nextInt();**

**// Creating the sales matrix**

**int sales[][] = new int[m][n];**

**// Taking input for sales data**

**System.out.println(&quot;Enter the sales data:&quot;);**

**for (int i = 0; i &lt; m; i++) {**

**System.out.println(&quot;Enter sales for year &quot; + (i + 1) + &quot;:&quot;);**

**for (int j = 0; j &lt; n; j++) {**

**System.out.print(&quot;Item &quot; + (j + 1) + &quot;: &quot;);**

**sales[i][j] = sc.nextInt();**

**}**

**}**

**// Finding the item with highest demand**

**int maxSales = sales[0][0];**

**int maxYear = 0, maxItem = 0;**

**for (int i = 0; i &lt; m; i++) {**

**for (int j = 0; j &lt; n; j++) {**

**if (sales[i][j] &gt; maxSales) {**

**maxSales = sales[i][j];**

**maxYear = i;**

**maxItem = j;**

**}**

**}**

**}**

**// Displaying the results**

**System.out.println(&quot;\nSales Data (Matrix Format):&quot;);**

**for (int i = 0; i &lt; m; i++) {**

**for (int j = 0; j &lt; n; j++) {**

**System.out.print(sales[i][j] + &quot;\t&quot;);**

**}**

**System.out.println();**

**}**

**System.out.println(&quot;\nItem with highest demand:&quot;);**

**System.out.println(&quot;Year: &quot; + (maxYear + 1) + &quot; | Item ID: &quot; + (maxItem + 1) + &quot; | Sales: &quot; +**

**maxSales);**

**sc.close();**

**}**

**}**

**Explanation**

**1. Input:**

**o m → Number of financial years.**

**o n → Number of items sold per year.**

**o A m x n matrix (sales[][]) stores sales data.**

**2. Processing:**

**o It loops through the matrix to find the highest sales value.**

**o It keeps track of the year (maxYear) and item (maxItem).**

**3. Output:**

**o Prints the sales matrix for reference.**

**o Displays the year, item ID, and sales count for the item with the highest**

**demand.**

**Example Execution**

**Input:**

**Enter the number of financial years (m): 3**

**Enter the number of items sold (n): 4**

**Enter sales for year 1:**

**Item 1: 10**

**Item 2: 15**

**Item 3: 30**

**Item 4: 20**

**Enter sales for year 2:**

**Item 1: 25**

**Item 2: 50**

**Item 3: 40**

**Item 4: 35**

**Enter sales for year 3:**

**Item 1: 5**

**Item 2: 60**

**Item 3: 20**

**Item 4: 45**

**Sales Data (Matrix Format)**

**10 15 30 20**

**25 50 40 35**

**5 60 20 45**

**Output:**

**Item with highest demand:**

**Year: 3 | Item ID: 2 | Sales: 60**

** Explanation:**

**o The highest sales value is 60, which occurs in Year 3 (row index 2) and Item 2**

**(column index 1).**

**Complexity Analysis**

** Time Complexity: O(m \* n) (scanning all elements once)**

** Space Complexity: O(m \* n) (for storing the matrix)**

**Improvements**

** If multiple items have the same highest sales, we can print all of them.**

** Can store item names for a more detailed output.**

**3.b)(OR)**

**import java.util.Scanner;**

**class Finance {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for matrix dimensions**

**System.out.print(&quot;Enter the number of years (n): &quot;);**

**int n = sc.nextInt();**

**System.out.print(&quot;Enter the number of items (m): &quot;);**

**int m = sc.nextInt();**

**int[][] sales = new int[n][m]; // 2D array for storing sales data**

**// Taking input for sales data**

**System.out.println(&quot;Enter the sales data:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**for (int j = 0; j &lt; m; j++) {**

**System.out.print(&quot;Sales for Year &quot; + (i + 1) + &quot;, Item &quot; + (j + 1) + &quot;: &quot;);**

**sales[i][j] = sc.nextInt();**

**}**

**}**

**// Finding the item with the highest sales for each year**

**for (int i = 0; i &lt; n; i++) {**

**int maxSales = sales[i][0]; // Assume first item has max sales**

**int itemId = 1; // ID of item with max sales**

**for (int j = 1; j &lt; m; j++) {**

**if (sales[i][j] &gt; maxSales) {**

**maxSales = sales[i][j];**

**itemId = j + 1; // Item IDs start from 1**

**}**

**}**

**System.out.println(&quot;For Year &quot; + (i + 1) + &quot;, Item &quot; + itemId + &quot; had the highest**

**sales: &quot; + maxSales);**

**}**

**sc.close();**

**}**

**}**

**Explanation**

**1. User Inputs Dimensions (n × m)**

**o n: Number of financial years.**

**o m: Number of items (IDs).**

**2. Input Sales Data**

**o The program reads sales data for each year and item.**

**3. Finds the Item with Maximum Sales for Each Year**

**o The program loops through each year (i).**

**o It finds the item with the highest sales in that year.**

**o The item ID is printed along with the highest sales value.**

**Example Output**

**Input:**

**Enter the number of years (n): 2**

**Enter the number of items (m): 3**

**Enter the sales data:**

**Sales for Year 1, Item 1: 500**

**Sales for Year 1, Item 2: 700**

**Sales for Year 1, Item 3: 650**

**Sales for Year 2, Item 1: 800**

**Sales for Year 2, Item 2: 600**

**Sales for Year 2, Item 3: 900**

**Matrix Representation**

**Year 1: [500, 700, 650]**

**Year 2: [800, 600, 900]**

**Output:**

**For Year 1, Item 2 had the highest sales: 700**

**For Year 2, Item 3 had the highest sales: 900**

**Key Learnings**

** Used a 2D array to store sales data.**

** Used nested loops to take input and find the highest-selling item.**

** Identified the item with maximum sales for each year.**

**This program efficiently identifies the most in-demand product for multiple years.**

**3. C.Write a program that accepts an ‘n’ ordered square matrix and calculates absolute**

**difference between the sums of elements in their diagonals.**

**import java.util.Scanner;**

**public class DiagonalDifference {**

**public static int diagonal Difference(int[][] matrix) {**

**int n = matrix.length;**

**int primaryDiagonal = 0;**

**int secondaryDiagonal = 0;**

**// Loop through the matrix to calculate diagonal sums**

**for (int i = 0; i &lt; n; i++) {**

**primaryDiagonal += matrix[i][i]; // Elements of primary diagonal**

**secondaryDiagonal += matrix[i][n - 1 - i]; // Elements of secondary diagonal**

**}**

**return Math.abs(primaryDiagonal - secondaryDiagonal);**

**}**

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

**Scanner scanner = new Scanner(System.in);**

**System.out.print(&quot;Enter the order of the square matrix (n): &quot;);**

**int n = scanner.nextInt();**

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

**System.out.println(&quot;Enter the matrix row by row:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**for (int j = 0; j &lt; n; j++) {**

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

**}**

**}**

**int result = diagonalDifference(matrix);**

**System.out.println(&quot;Absolute diagonal difference: &quot; + result);**

**scanner.close();**

**}**

**}**

**Explanation of the Java Program:**

**1. Method diagonalDifference(int[][] matrix):**

**o This method calculates the absolute difference between the two diagonals.**

**o It iterates through the matrix:**

** The primary diagonal consists of elements matrix[i][i].**

** The secondary diagonal consists of elements matrix[i][n - 1 - i].**

**o The absolute difference is returned.**

**2. main Method:**

**o Reads an integer n from the user, representing the size of the matrix.**

**o Accepts n x n elements row-wise as input.**

**o Calls diagonalDifference(matrix) to compute the difference.**

**o Displays the result.**

**(Or)**

**3.c .import java.util.Scanner;**

**class DiagonalDifference {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for matrix size**

**System.out.print(&quot;Enter the size of the square matrix (n): &quot;);**

**int n = sc.nextInt();**

**int[][] a = new int[n][n]; // 2D array for matrix**

**int d1 = 0, d2 = 0; // Variables to store diagonal sums**

**// Taking input for matrix elements**

**System.out.println(&quot;Enter the matrix elements:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**for (int j = 0; j &lt; n; j++) {**

**a[i][j] = sc.nextInt();**

**}**

**}**

**// Calculating sums of diagonals**

**for (int i = 0; i &lt; n; i++) {**

**d1 += a[i][i]; // Primary diagonal sum**

**d2 += a[i][n - i - 1]; // Secondary diagonal sum**

**}**

**// Calculating absolute difference**

**int diff = Math.abs(d1 - d2);**

**// Output the result**

**System.out.println(&quot;Absolute difference between diagonals: &quot; + diff);**

**sc.close();**

**}**

**}**

**Explanation**

**1. User Inputs Matrix Size (n)**

**o The program reads an integer n representing the size of the n × n square matrix.**

**2. Input Matrix Elements**

**o The user enters n × n matrix elements row by row.**

**3. Calculate the Sum of Two Diagonals**

**o Primary Diagonal (d1): Elements where i == j**

** Example: a[0][0], a[1][1], a[2][2], etc.**

**o Secondary Diagonal (d2): Elements where i + j == n - 1**

** Example: a[0][2], a[1][1], a[2][0], etc.**

**4. Compute Absolute Difference**

**o Math.abs(d1 - d2) is used to ensure a positive result.**

**Example Output**

**Input:**

**Enter the size of the square matrix (n): 3**

**Enter the matrix elements:**

**11 2 4**

**4 5 6**

**10 8 -12**

**Matrix Representation**

**11 2 4**

**4 5 6**

**10 8 -12**

**Calculations**

** Primary Diagonal Sum (d1) = 11 + 5 + (-12) = 4**

** Secondary Diagonal Sum (d2) = 4 + 5 + 10 = 19**

** Absolute Difference = |4 - 19| = 15**

**Output:**

**Absolute difference between diagonals: 15**

**Key Takeaways**

**✅ Uses a 2D array to store the square matrix.**

**✅ Single loop to compute both diagonals efficiently.**

**✅ Math.abs() ensures the absolute difference is always positive.**

**This program efficiently computes the diagonal difference for any square matrix.**

**4.a Create a class Box that uses a parameterized constructor to**

**initialize the dimensions of a box. The dimensions of the Box are**

**width, height, depth. The class should have a method that can return**

**the volume of the box. Create an object of the Box class and test the**

**functionalities.**

**import java.util.Scanner;**

**class Box {**

**int width, height, depth;**

**// Parameterized Constructor**

**Box(int wid, int ht, int dpt) {**

**this.width = wid;**

**this.height = ht;**

**this.depth = dpt;**

**}**

**// Method to calculate volume**

**int volume() {**

**return width \* height \* depth;**

**}**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input from the user**

**System.out.print(&quot;Enter width: &quot;);**

**int wid = sc.nextInt();**

**System.out.print(&quot;Enter height: &quot;);**

**int ht = sc.nextInt();**

**System.out.print(&quot;Enter depth: &quot;);**

**int dpt = sc.nextInt();**

**// Creating an object of Box class**

**Box myBox = new Box(wid, ht, dpt);**

**// Displaying the volume**

**System.out.println(&quot;Volume of the box: &quot; + myBox.volume());**

**sc.close();**

**}**

**}**

**Explanation**

**1. Class Definition (Box)**

**o The class has three instance variables: width, height, and depth.**

**o A parameterized constructor initializes these values.**

**2. Method to Calculate Volume (volume())**

**o This method returns width \* height \* depth, which gives the volume of the**

**box.**

**3. Main Method (main)**

**o Takes user input for width, height, and depth.**

**o Creates an object of the Box class.**

**o Calls the volume() method and displays the result.**

**Example Output**

**Input:**

**Enter width: 5**

**Enter height: 10**

**Enter depth: 3**

**Calculations**

**Volume=5×10×3=150Volume = 5 \times 10 \times 3 = 150Volume=5×10×3=150**

**Output:**

**Volume of the box: 150**

**Key Features**

**✅ Uses a parameterized constructor to initialize object values.**

**✅ Encapsulation of logic with a dedicated volume() method.**

**✅ Efficient memory handling by directly storing input in object attributes.**

**This program effectively demonstrates object-oriented programming (OOP) in Java using**

**constructors**

**4.b Create a new class called Calculator with the following methods:**

**● A static method called powering (int num1,int num2)**

**● This method should return num1 to the power num2.**

**● A static method called powerDouble(double num1,double num2).**

**● This method should return num1 to the power num2.**

**● Invoke both the methods and test the functionalities. Also count number of objects created.**

**import java.lang.Math;**

**class Calculator {**

**// Static variable to count the number of objects created**

**static int objectCount = 0;**

**// Constructor to increment the object count**

**Calculator() {**

**objectCount++;**

**}**

**// Static method for integer power calculation**

**static int powering(int num1, int num2) {**

**return (int) Math.pow(num1, num2);**

**}**

**// Static method for double power calculation**

**static double powerDouble(double num1, double num2) {**

**return Math.pow(num1, num2);**

**}**

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

**// Creating objects**

**Calculator calc1 = new Calculator();**

**Calculator calc2 = new Calculator();**

**// Testing integer power method**

**int result1 = Calculator.powering(3, 4); // 3^4 = 81**

**System.out.println(&quot;3^4 = &quot; + result1);**

**// Testing double power method**

**double result2 = Calculator.powerDouble(2.5, 3.2); // 2.5^3.2**

**System.out.println(&quot;2.5^3.2 = &quot; + result2);**

**// Display the number of objects created**

**System.out.println(&quot;Number of Calculator objects created: &quot; + Calculator.objectCount);**

**}**

**}**

**Explanation**

**1. Class Calculator**

**o Contains a static variable objectCount to track how many objects are created.**

**o The constructor increments the object count whenever a new object is created.**

**2. Static Methods**

**o powering(int num1, int num2): Uses Math.pow() to calculate num1^num2**

**and returns an integer value.**

**o powerDouble(double num1, double num2): Uses Math.pow() to compute**

**num1^num2 with double precision.**

**3. Main Method**

**o Creates two objects of Calculator, increasing objectCount to 2.**

**o Calls powering(3,4) → Returns 81.**

**o Calls powerDouble(2.5,3.2) → Returns 18.767569 (approx).**

**o Displays the number of objects created.**

**Example Output**

**3^4 = 81**

**2.5^3.2 = 18.767569**

**Number of Calculator objects created: 2**

**Key Features**

**✅ Uses Static Methods → No need to create objects to call power functions.**

**✅ Uses Math.pow() → Handles power calculations efficiently.**

**✅ Tracks Object Creation → Demonstrates the use of static variables.**

**5) A) Accept a String and a number ‘n’ from user. Divide the given string into**

**substrings each of size ‘n’ and sort them lexicographically.**

**import java.util.\*;**

**class StringSorter {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input from user**

**System.out.print(&quot;Enter the string: &quot;);**

**String s = sc.nextLine();**

**System.out.print(&quot;Enter the value of n: &quot;);**

**int n = sc.nextInt();**

**// Checking if &#39;n&#39; is valid**

**if (n &lt;= 0 || n &gt; s.length()) {**

**System.out.println(&quot;Invalid input! &#39;n&#39; should be between 1 and the**

**length of the string.&quot;);**

**return;**

**}**

**// Calculating number of substrings**

**int numSubstrings = s.length() - n + 1;**

**String[] substrings = new String[numSubstrings];**

**// Extracting substrings of size &#39;n&#39;**

**for (int i = 0; i &lt; numSubstrings; i++) {**

**substrings[i] = s.substring(i, i + n);**

**}**

**// Sorting substrings lexicographically**

**Arrays.sort(substrings);**

**// Displaying the sorted substrings**

**System.out.println(&quot;Sorted substrings: &quot; + Arrays.toString(substrings));**

**sc.close();**

**}**

**}**

**Explanation**

**1. User Input:**

**o The user enters a string and an integer n.**

**2. Validation:**

**o The program checks if n is valid (between 1 and the string length).**

**3. Extracting Substrings:**

**o The total number of substrings of size n is s.length() - n + 1.**

**o A loop extracts substrings and stores them in an array.**

**4. Sorting:**

**o Arrays.sort(substrings) is used to sort the substrings lexicographically.**

**5. Output:**

**o The sorted substrings are displayed.**

**Example Output**

**Input:**

**Enter the string: applebanana**

**Enter the value of n: 3**

**Processing:**

** Generated substrings: [&quot;app&quot;, &quot;ppl&quot;, &quot;ple&quot;, &quot;leb&quot;, &quot;eba&quot;, &quot;ban&quot;, &quot;ana&quot;,**

**&quot;nan&quot;, &quot;ana&quot;]**

** Sorted order: [&quot;ana&quot;, &quot;ana&quot;, &quot;app&quot;, &quot;ban&quot;, &quot;eba&quot;, &quot;leb&quot;, &quot;nan&quot;, &quot;ple&quot;,**

**&quot;ppl&quot;]**

**Output:**

**Sorted substrings: [ana, ana, app, ban, eba, leb, nan, ple, ppl]**

**Enter the string: sravani**

**Enter the value of n: 4**

**Processing:**

**The program extracts all possible substrings of size n = 4 from &quot;sravani&quot;.**

**Step 1: Extract Substrings**

**We extract substrings of length 4 from &quot;sravani&quot;:**

**1. &quot;srav&quot; (index 0-3)**

**2. &quot;rava&quot; (index 1-4)**

**3. &quot;avan&quot; (index 2-5)**

**4. &quot;vani&quot; (index 3-6)**

**So, the substrings array before sorting:**

**[&quot;srav&quot;, &quot;rava&quot;, &quot;avan&quot;, &quot;vani&quot;]**

**Step 2: Sort Substrings Lexicographically**

**Sorting in dictionary order:**

**[&quot;avan&quot;, &quot;rava&quot;, &quot;srav&quot;, &quot;vani&quot;]**

**Final Output:**

**Sorted substrings: [avan, rava, srav, vani]**

**5. B) Accept Array of strings and display the number of ovals and consonants**

**occurred in each string.**

**import java.util.\*;**

**public class VowelConsonantCounter {**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for number of strings**

**System.out.print(&quot;Enter the number of strings: &quot;);**

**int numStrings = sc.nextInt();**

**sc.nextLine(); // Consume the newline character**

**String[] strings = new String[numStrings];**

**// Taking input for each string**

**for (int i = 0; i &lt; numStrings; i++) {**

**System.out.print(&quot;Enter string &quot; + (i + 1) + &quot;: &quot;);**

**strings[i] = sc.nextLine();**

**}**

**// Processing each string**

**for (String str : strings)**

**{**

**int vowels = 0, consonants = 0;**

**// Convert string to lowercase to make comparisons easier**

**str = str.toLowerCase();**

**for (char ch : str.toCharArray())**

**{**

**if (Character.isLetter(ch))**

**{ // Check if it&#39;s a letter**

**if (&quot;aeiou&quot;.indexOf(ch) != -1) {**

**vowels++;**

**} else {**

**consonants++;**

**}**

**}**

**}**

**// Display results for each string**

**System.out.println(&quot;String: &quot; + str);**

**System.out.println(&quot;Vowels: &quot; + vowels + &quot;, Consonants: &quot; + consonants);**

**System.out.println(&quot;---------------------------------&quot;);**

**}**

**sc.close();**

**}**

**}**

**Explanation:**

**1. Takes an array of strings from the user.**

**2. Iterates through each string, counting vowels (a, e, i, o, u) and consonants.**

**3. Ignores non-alphabetic characters (digits, spaces, punctuation).**

**4. Prints the results for each string.**

**Example Output:**

**Input:**

**Enter the number of strings: 3**

**Enter string 1: hello**

**Enter string 2: world**

**Enter string 3: programming**

**Processing:**

**1. &quot;hello&quot; → Vowels = 2 (e, o), Consonants = 3 (h, l, l)**

**2. &quot;world&quot; → Vowels = 1 (o), Consonants = 4 (w, r, l, d)**

**3. &quot;programming&quot; → Vowels = 3 (o, a, i), Consonants = 8 (p, r, g, r, m, m, n, g)**

**Output:**

**String: hello**

**Vowels: 2, Consonants: 3**

**---------------------------------**

**String: world**

**Vowels: 1, Consonants: 4**

**---------------------------------**

**String: programming**

**Vowels: 3, Consonants: 8**

**---------------------------------**

**5.c. Accept ‘n’ number of strings and reverse the strings in alternate position**

**and display as single string in the order of the input acceptance.**

**import java.util.\*;**

**public class ReverseAlternateStrings {**

**// Function to reverse a given string**

**public static String reverse(String word) {**

**StringBuilder reversed = new StringBuilder(word);**

**return reversed.reverse().toString();**

**}**

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

**Scanner sc = new Scanner(System.in);**

**// Taking input for number of strings**

**System.out.print(&quot;Enter the number of strings: &quot;);**

**int n = sc.nextInt();**

**sc.nextLine(); // Consume the newline character**

**String[] strings = new String[n];**

**StringBuilder result = new StringBuilder();**

**// Taking input for each string**

**System.out.println(&quot;Enter the strings:&quot;);**

**for (int i = 0; i &lt; n; i++) {**

**strings[i] = sc.nextLine();**

**}**

**// Processing alternate strings**

**for (int i = 0; i &lt; n; i++) {**

**if (i % 2 == 0) {**

**result.append(strings[i]); // Keep even index strings as they are**

**} else {**

**result.append(reverse(strings[i])); // Reverse odd index strings**

**}**

**}**

**// Display the final single string**

**System.out.println(&quot;Final output string: &quot; + result.toString());**

**sc.close();**

**}**

**}**

**Explanation**

**1. Reverse Function (reverse(String word))**

**o Uses StringBuilder.reverse() to efficiently reverse the string.**

**2. User Input:**

**o The user enters the number of strings (n).**

**o Then n strings are taken as input and stored in an array.**

**3. Processing the Strings:**

**o Loops through the array.**

**o If the index is even (0, 2, 4...), the string remains unchanged.**

**o If the index is odd (1, 3, 5...), the string is reversed using the reverse()**

**function.**

**4. Concatenation:**

**o The final modified strings are appended into a single result string.**

**5. Final Output:**

**o The program displays the concatenated string.**

**Example Output**

**Input:**

**Enter the number of strings: 5**

**Enter the strings:**

**apple**

**banana**

**cherry**

**date**

**elderberry**

**Processing:**

** &quot;apple&quot; → kept as &quot;apple&quot;**

** &quot;banana&quot; → reversed to &quot;ananab&quot;**

** &quot;cherry&quot; → kept as &quot;cherry&quot;**

** &quot;date&quot; → reversed to &quot;etad&quot;**

** &quot;elderberry&quot; → kept as &quot;elderberry&quot;**

**Output:**

**Final output string: appleananabcherryetadelderberry**

**Corrections &amp; Improvements:**

**1. Fixed Input Handling:**

**o Used sc.nextLine(); properly after sc.nextInt(); to avoid skipped input.**

**2. Efficient String Reversal:**

**o Used StringBuilder.reverse() instead of manual reversal for better**

**performance.**

**3. Efficient Concatenation:**

**o Used StringBuilder.append() instead of += to handle large inputs efficiently.**

**5.d. Accept two strings from the user and**

**determine if the strings are anagrams or**

**not.**

**An anagram is a word or phrase formed by rearranging the letters of another word or phrase.**

**The letters must be used exactly as they appear, without adding or removing any characters.**

**Examples of Anagrams:**

**1. &quot;listen&quot; → &quot;silent&quot;**

**2. &quot;earth&quot; → &quot;heart&quot;**

**3. &quot;evil&quot; → &quot;vile&quot;**

**4. &quot;dusty&quot; → &quot;study&quot;**

**Both words in an anagram pair must have:**

** The same letters.**

** The same frequency of each letter.**

** The same length.**

**import java.util.Arrays;**

**import java.util.Scanner;**

**public class AnagramChecker {**

**public static boolean areAnagrams(String str1, String str2) {**

**// Remove spaces and convert to lowercase for case-insensitive**

**comparison**

**str1 = str1.replaceAll(&quot;\\s&quot;, &quot;&quot;).toLowerCase();**

**str2 = str2.replaceAll(&quot;\\s&quot;, &quot;&quot;).toLowerCase();**

**// If lengths are not equal, they cannot be anagrams**

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

**return false;**

**}**

**// Convert to character arrays and sort**

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

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

**Arrays.sort(charArray1);**

**Arrays.sort(charArray2);**

**// Compare sorted arrays**

**return Arrays.equals(charArray1, charArray2);**

**}**

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

**Scanner sc = new Scanner(System.in);**

**// Accepting two strings from the user**

**System.out.print(&quot;Enter first string: &quot;);**

**String str1 = sc.nextLine();**

**System.out.print(&quot;Enter second string: &quot;);**

**String str2 = sc.nextLine();**

**// Checking if the strings are anagrams**

**if (areAnagrams(str1, str2)) {**

**System.out.println(&quot;The strings are anagrams.&quot;);**

**} else {**

**System.out.println(&quot;The strings are NOT anagrams.&quot;);**

**}**

**sc.close();**

**}**

**}**

**Explanation**

**1. Remove Spaces &amp; Convert to Lowercase**

**o This ensures the check is case-insensitive and ignores spaces.**

**2. Check Length**

**o If the two strings have different lengths, they cannot be anagrams.**

**3. Sort the Characters**

**o Both strings are converted to character arrays and sorted.**

**4. Compare the Sorted Arrays**

**o If the sorted character arrays are equal, the words are anagrams.**

**Example Output**

**Case 1: Anagrams**

**Enter first string: listen**

**Enter second string: silent**

**The strings are anagrams.**

**Case 2: Not Anagrams**

**Enter first string: hello**

**Enter second string: world**

**The strings are NOT anagrams.**

**7. Packages: Student, Sports, and Report**

For this program, you need to create a specific folder structure.

1. Create a main project folder.
2. Inside it, create three sub-folders: studentpackage, sportspackage, and reportpackage.
3. Save each file in its corresponding folder.

**File 1: studentpackage/Student.java**

package studentpackage;

public class Student {

private String name;

private int rollNumber;

public Student(String name, int rollNumber) {

this.name = name;

this.rollNumber = rollNumber;

}

public String getName() {

return name;

}

public int getRollNumber() {

return rollNumber;

}

}

**File 2: sportspackage/Sports.java**

package sportspackage;

public interface Sports {

String getSportName();

int getSportScore();

}

**File 3: reportpackage/Report.java**

package reportpackage;

import studentpackage.Student;

import sportspackage.Sports;

public class Report implements Sports {

private Student student;

private String sportName;

private int sportScore;

public Report(Student student, String sportName, int sportScore) {

this.student = student;

this.sportName = sportName;

this.sportScore = sportScore;

}

@Override

public String getSportName() {

return sportName;

}

@Override

public int getSportScore() {

return sportScore;

}

public void generateReport() {

System.out.println("----- Student Report -----");

System.out.println("Name : " + student.getName());

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

System.out.println("Sport : " + getSportName());

System.out.println("Score : " + getSportScore());

}

public static void main(String[] args) {

Student student = new Student("Alice", 101);

Report report = new Report(student, "Basketball", 85);

report.generateReport();

}

}

**8.A) Integer Division with Exception Handling**

**File Name: IntegerDivision.java**

import java.util.Scanner;

public class IntegerDivision {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter first number (Num1): ");

String input1 = scanner.nextLine();

System.out.print("Enter second number (Num2): ");

String input2 = scanner.nextLine();

int num1 = Integer.parseInt(input1);

int num2 = Integer.parseInt(input2);

int result = num1 / num2;

System.out.println("Result of " + num1 + " / " + num2 + " = " + result);

} catch (NumberFormatException e) {

System.out.println("Error: Please enter valid integers only.");

} catch (ArithmeticException e) {

System.out.println("Error: Cannot divide by zero.");

} finally {

scanner.close();

}

}

}

**8.B) User-Defined Exception**

**File Name: User.java**

import java.util.\*;

class MyExp extends Exception {

public MyExp(String str) {

super(str);

}

}

public class User {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

String n1, n2;

try {

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

n1 = sc.nextLine();

int a = Integer.parseInt(n1);

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

n2 = sc.nextLine();

int b = Integer.parseInt(n2);

if (b <= 0) {

throw new MyExp("Arithmetic Exception: Division by zero or negative number is not allowed.");

}

System.out.println("Division: " + (a / b));

} catch (NumberFormatException e) {

System.out.println("Error: Please enter only integers.");

} catch (MyExp e) {

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

} finally {

sc.close();

}

}

}

**9.A) Multithreading by Extending Thread**

**File Name: MultiThreadExample.java**

class Thread1 extends Thread {

public void run() {

for (int i = 1; i <= 6; i++) {

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println("Good morning");

}

}

}

class Thread2 extends Thread {

public void run() {

for (int i = 1; i <= 6; i++) {

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println("Hello");

}

}

}

class Thread3 extends Thread {

public void run() {

for (int i = 1; i <= 6; i++) {

try {

Thread.sleep(3000);

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println("Welcome");

}

}

}

public class MultiThreadExample {

public static void main(String[] args) {

Thread1 t1 = new Thread1();

Thread2 t2 = new Thread2();

Thread3 t3 = new Thread3();

t1.start();

t2.start();

t3.start();

}

}

**9.B) Thread Synchronization**

**File Name: SyncExample.java**

class Printer {

synchronized void print(String message) {

for (int i = 1; i <= 3; i++) {

System.out.println(message + " - " + i);

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

class PrintJob extends Thread {

Printer printer;

String message;

PrintJob(Printer printer, String message) {

this.printer = printer;

this.message = message;

}

public void run() {

printer.print(message);

}

}

public class SyncExample {

public static void main(String[] args) {

Printer sharedPrinter = new Printer();

PrintJob t1 = new PrintJob(sharedPrinter, "Thread 1 printing");

PrintJob t2 = new PrintJob(sharedPrinter, "Thread 2 printing");

t1.start();

t2.start();

}

}

**10.A) Swing Registration Page**

**File Name: Registration.java**

import javax.swing.\*;

import java.awt.\*;

public class Registration {

public static void main(String[] args) {

JFrame frame = new JFrame("Registration Form");

frame.setSize(400, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(new BorderLayout());

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(6, 2, 10, 10));

JLabel nameLabel = new JLabel("Enter your name:");

JTextField nameField = new JTextField(20);

JLabel ageLabel = new JLabel("Enter your age:");

JTextField ageField = new JTextField(20);

JLabel emailLabel = new JLabel("Enter your email:");

JTextField emailField = new JTextField(20);

JLabel passwordLabel = new JLabel("Enter your password:");

JPasswordField passwordField = new JPasswordField(20);

JLabel confirmPasswordLabel = new JLabel("Confirm password:");

JPasswordField confirmPasswordField = new JPasswordField(20);

JButton submitButton = new JButton("Register");

panel.add(nameLabel);

panel.add(nameField);

panel.add(ageLabel);

panel.add(ageField);

panel.add(emailLabel);

panel.add(emailField);

panel.add(passwordLabel);

panel.add(passwordField);

panel.add(confirmPasswordLabel);

panel.add(confirmPasswordField);

panel.add(new JLabel());

panel.add(submitButton);

frame.add(panel, BorderLayout.CENTER);

frame.setVisible(true);

}

}

**10.B) Mouse Events with Adapter Classes**

**File Name: MouseEventDemo.java**

import java.awt.\*;

import java.awt.event.\*;

public class MouseEventDemo extends Frame {

Label label;

MouseEventDemo() {

label = new Label();

label.setBounds(20, 50, 100, 20);

add(label);

addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

label.setText("Mouse Clicked");

}

public void mouseEntered(MouseEvent e) {

label.setText("Mouse Entered");

}

public void mouseExited(MouseEvent e) {

label.setText("Mouse Exited");

}

public void mouseReleased(MouseEvent e) {

label.setText("Mouse Released");

}

});

setSize(300, 300);

setLayout(null);

setVisible(true);

}

public static void main(String[] args) {

new MouseEventDemo();

}

}

**12.A) Registration Validation**

**File Name: Registration3.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Registration3 extends JFrame implements ActionListener {

JLabel l1, l2, l3, l4;

JTextField tf1, tf2;

JButton btn1;

JPasswordField pf1, pf2;

Registration3() {

setVisible(true);

setSize(700, 700);

setLayout(new BorderLayout());

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setTitle("Registration Form");

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(30, 30));

l1 = new JLabel("Name");

tf1 = new JTextField();

panel.add(l1);

panel.add(tf1);

l2 = new JLabel("Email");

tf2 = new JTextField();

panel.add(l2);

panel.add(tf2);

btn1 = new JButton("Submit");

btn1.addActionListener(this);

panel.add(btn1);

add(panel);

}

public void actionPerformed(ActionEvent e) {

String name = tf1.getText();

String email = tf2.getText();

if (name.isEmpty() || email.isEmpty()) {

JOptionPane.showMessageDialog(this, "Error: All fields must be filled.");

} else {

JOptionPane.showMessageDialog(this, "Success: All fields are filled!");

}

}

public static void main(String[] args) {

new Registration3();

}

}

**12.B) Color Sliders**

**File Name: ColorSlider.java**

import java.awt.\*;

import javax.swing.\*;

import javax.swing.event.\*;

public class ColorSlider extends JFrame implements ChangeListener {

private JSlider redSlider, greenSlider, blueSlider;

private Container contentPane;

public ColorSlider() {

super("Color Slider");

contentPane = getContentPane();

contentPane.setLayout(new FlowLayout());

redSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);

greenSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);

blueSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);

redSlider.addChangeListener(this);

greenSlider.addChangeListener(this);

blueSlider.addChangeListener(this);

contentPane.add(new JLabel("Red"));

contentPane.add(redSlider);

contentPane.add(new JLabel("Green"));

contentPane.add(greenSlider);

contentPane.add(new JLabel("Blue"));

contentPane.add(blueSlider);

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

public void stateChanged(ChangeEvent e) {

int r = redSlider.getValue();

int g = greenSlider.getValue();

int b = blueSlider.getValue();

Color color = new Color(r, g, b);

contentPane.setBackground(color);

}

public static void main(String[] args) {

new ColorSlider();

}

}

**13.A) Custom ArrayList**

**File Name: MyArrayListDemo.java**

class MyArrayList<T> {

private Object[] data;

private int size;

private static final int INITIAL\_CAPACITY = 5;

public MyArrayList() {

data = new Object[INITIAL\_CAPACITY];

size = 0;

}

public void add(T element) {

if (size == data.length) {

resize();

}

data[size++] = element;

}

public T get(int index) {

checkIndex(index);

return (T) data[index];

}

public void remove(int index) {

checkIndex(index);

for (int i = index; i < size - 1; i++) {

data[i] = data[i + 1];

}

data[size - 1] = null;

size--;

}

public int size() {

return size;

}

private void resize() {

int newCapacity = data.length \* 2;

Object[] newData = new Object[newCapacity];

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

newData[i] = data[i];

}

data = newData;

}

private void checkIndex(int index) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Index " + index + " out of bounds");

}

}

}

public class MyArrayListDemo {

public static void main(String[] args) {

MyArrayList<Integer> list = new MyArrayList<>();

System.out.println("Adding elements:");

list.add(10);

list.add(20);

list.add(30);

list.add(40);

list.add(50);

list.add(60);

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

System.out.println("Element at index " + i + ": " + list.get(i));

}

System.out.println("\nRemoving element at index 2 (value: 30):");

list.remove(2);

System.out.println("\nElements after removal:");

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

System.out.println("Element at index " + i + ": " + list.get(i));

}

System.out.println("\nCurrent size of list: " + list.size());

}

}

**13.B) Employee HashMap**

**File Name: EmployeeDemo.java**

import java.util.HashMap;

import java.util.Scanner;

class Employee {

private int id;

private String name;

private String department;

private double salary;

public Employee(int id, String name, String department, double salary) {

this.id = id;

this.name = name;

this.department = department;

this.salary = salary;

}

public void setName(String name) {

this.name = name;

}

public void setDepartment(String department) {

this.department = department;

}

public void setSalary(double salary) {

this.salary = salary;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public String getDepartment() {

return department;

}

public double getSalary() {

return salary;

}

public void displayEmployee() {

System.out.println("ID: " + id);

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

System.out.println("Department: " + department);

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

}

}

public class EmployeeDemo {

public static void main(String[] args) {

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

Scanner scanner = new Scanner(System.in);

System.out.print("How many employees you want to add? ");

int n = scanner.nextInt();

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

System.out.println("\nEnter details for Employee " + (i + 1) + ":");

System.out.print("ID: ");

int id = scanner.nextInt();

scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Department: ");

String department = scanner.nextLine();

System.out.print("Salary: ");

double salary = scanner.nextDouble();

Employee emp = new Employee(id, name, department, salary);

employeeMap.put(id, emp);

}

System.out.print("\nEnter Employee ID to search: ");

int searchId = scanner.nextInt();

if (employeeMap.containsKey(searchId)) {

System.out.println("\nEmployee Found:");

employeeMap.get(searchId).displayEmployee();

} else {

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

}

scanner.close();

}

}

**14.A) File Info and Content Display**

**File Name: FileInfoDisplay.java**

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.util.Scanner;

public class FileInfoDisplay {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the file name with path: ");

String fileName = scanner.nextLine();

File file = new File(fileName);

if (file.exists()) {

System.out.println("File exists: Yes");

System.out.println("Readable: " + file.canRead());

System.out.println("Writable: " + file.canWrite());

System.out.println("File type: " + (file.isFile() ? "Regular file" : "Directory"));

System.out.println("File length (bytes): " + file.length());

System.out.println("\n--- File Content ---");

try (FileInputStream fis = new FileInputStream(file)) {

int ch;

while ((ch = fis.read()) != -1) {

System.out.print((char) ch);

}

} catch (IOException e) {

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

}

} else {

System.out.println("File does not exist.");

}

scanner.close();

}

}

**14.B) Copy File using Character Streams**

**File Name: FileCopyCharacterStream.java**

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileCopyCharacterStream {

public static void main(String[] args) {

String sourceFile = "source.txt";

String destinationFile = "destination.txt";

FileReader reader = null;

FileWriter writer = null;

try {

reader = new FileReader(sourceFile);

writer = new FileWriter(destinationFile);

int ch;

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

writer.write(ch);

}

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

} catch (IOException e) {

System.out.println("An error occurred: " + e.getMessage());

} finally {

try {

if (reader != null)

reader.close();

if (writer != null)

writer.close();

} catch (IOException e) {

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

}

}

}

}

**14.C) Sum Integers with StringTokenizer**

**File Name: SumOfIntegers.java**

import java.util.Scanner;

import java.util.StringTokenizer;

public class SumOfIntegers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a line of integers separated by spaces:");

String inputLine = scanner.nextLine();

StringTokenizer tokenizer = new StringTokenizer(inputLine);

int sum = 0;

System.out.println("\nIntegers found:");

while (tokenizer.hasMoreTokens()) {

String token = tokenizer.nextToken();

try {

int number = Integer.parseInt(token);

System.out.println(number);

sum += number;

} catch (NumberFormatException e) {

System.out.println("Skipping non-integer token: " + token);

}

}

System.out.println("\nSum of all integers: " + sum);

scanner.close();

}

}

**15.A) Vector and Wrapper Classes**

**File Name: VectorDemo.java**

import java.util.\*;

public class VectorDemo {

public static void main(String[] args) {

int n;

Vector<Integer> v = new Vector<>();

Scanner s = new Scanner(System.in);

for (int j = 0; j < 10; j++) {

System.out.print("Enter a number --> ");

n = s.nextInt();

v.add(n);

}

System.out.println("Vector contents: " + v);

System.out.print("Enter the first occurrence of a number to remove --> ");

n = s.nextInt();

boolean removed = v.remove(Integer.valueOf(n));

System.out.println("First occurrence of element " + n + " is removed: " + removed);

System.out.println("Vector contents after remove operation: " + v);

System.out.print("Enter an index of an element to be removed --> ");

n = s.nextInt();

if (n >= 0 && n < v.size()) {

System.out.println("Removed element at index " + n + ": " + v.remove(n));

} else {

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

}

System.out.println("Final Vector contents: " + v);

}

}

**File Name: WrapperDemo.java**

public class WrapperDemo {

public static void main(String args[]) {

Integer a = new Integer(10);

int b = a.intValue();

int c = a;

System.out.println("int value of Integer obj b is " + b);

System.out.println("int value of Integer obj c is " + c);

int a1 = 10;

Integer b1 = Integer.valueOf(a1);

Integer c1 = a;

if (b1 instanceof Integer)

System.out.println("TRUE b1 is an instance of Integer and value is " + b1);

if (c1 instanceof Integer)

System.out.println("TRUE c1 is an instance of Integer and value is " + c1);

}

}

**15.B) Generate Random Numbers**

**File Name: RandomBetweenTwoNumbers.java**

import java.util.Scanner;

import java.util.Random;

public class RandomBetweenTwoNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Random random = new Random();

System.out.print("Enter the lower bound (x1 > 0): ");

int x1 = scanner.nextInt();

System.out.print("Enter the upper bound (x2): ");

int x2 = scanner.nextInt();

if (x1 <= 0 || x1 >= x2) {

System.out.println("Invalid input! Make sure x1 > 0 and x1 < x2.");

return;

}

System.out.print("Enter how many random numbers to generate: ");

int count = scanner.nextInt();

System.out.println("\nRandom numbers between " + x1 + " and " + x2 + ":");

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

int rand = random.nextInt(x2 - x1 + 1) + x1;

System.out.println(rand);

}

}

}

**15.C) Client-Server Application**

**File Name: CircleServer.java**

import java.io.\*;

import java.net.\*;

public class CircleServer {

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

ServerSocket serverSocket = new ServerSocket(5000);

System.out.println("Server is running and waiting for client...");

Socket socket = serverSocket.accept();

System.out.println("Client connected.");

DataInputStream in = new DataInputStream(socket.getInputStream());

DataOutputStream out = new DataOutputStream(socket.getOutputStream());

double radius = in.readDouble();

System.out.println("Received radius from client: " + radius);

double area = Math.PI \* radius \* radius;

out.writeDouble(area);

System.out.println("Sent area to client: " + area);

socket.close();

serverSocket.close();

}

}

**File Name: CircleClient.java**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class CircleClient {

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

Socket socket = new Socket("localhost", 5000);

DataInputStream in = new DataInputStream(socket.getInputStream());

DataOutputStream out = new DataOutputStream(socket.getOutputStream());

Scanner scanner = new Scanner(System.in);

System.out.print("Enter radius of circle: ");

double radius = scanner.nextDouble();

out.writeDouble(radius);

double area = in.readDouble();

System.out.println("Area of circle received from server: " + area);

socket.close();

}

}