**JAVA LAB PROGRAMS**

1. **Hello World**

public class Main {

public static void main(String[] args) {

System.out.println("Hello World!");

}

}

1. **Check positive or negative (if statements)**

public class NumberCheck {

public static void main(String[] args) {

int n=-10;

if(n>10)

System.out.println(n + " is positive");

else if(n<0)

System.out.println(n + " is negative");

else

System.out.println(n + " is zero");

}

}

1. **Check Leap year (if statements)**

public class LeapYear {

public static void main(String[] args) {

int yr=2100;

if(yr%100==0){

if(yr%400==0){

System.out.println(yr + " is a leap year");

}

else{

System.out.println(yr + " is not a leap year");

}

}

else{

if(yr%4==0){

System.out.println(yr + " is a leap year");

}

else{

System.out.println(yr + " is not a leap year");

}

}

}

}

1. **Numbers from 1 to 10 (for loop)**

public class PrintingNumbers {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

1. **Fibonacci series (for loop)**

public class Fibonacci {

public static void main(String[] args) {

int n=10;

int a=0,b=1;

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

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

for (int i = 0; i < n-2; i++) {

int c = a + b;

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

a = b;

b = c;

}

}

}

1. **Armstrong Number (while loop)**

public class ArmstrongNumber {

public static void main(String[] args) {

int n = 153;

int num = n;

int sum = 0;

while(num>0){

int r = num%10;

sum += r\*r\*r;

num = num/10;

}

if(sum==n)

System.out.println(n + " is a armstrong number");

else

System.out.println(n + " is not a armstrong number");

}

}

1. **LCM and HCF (while loop)**

public class LcmHcf {

public static void main(String[] args) {

int a=100;

int b=10;

int p=a\*b;

while(a%b!=0){

int r=a%b;

a=b;

b=r;

}

System.out.println("HCF is " + b);

System.out.println("LCM is " + (p/b));

}

}

1. **Reverse of a number (do while loop)**

public class ReverseOfANumber {

public static void main(String[] args) {

int num = 123;

int n = num;

int rev=0;

do{

int r = num%10;

rev = rev\*10+r;

num = num/10;

}while (num>0);

System.out.println("The reverse of " + n + " is " + rev);

}

}

1. **Sum of digits of a number (do while loop)**

public class SumOfDigits {

public static void main(String[] args) {

int num=123;

int n=num;

int sum=0;

do{

int r=num%10;

sum = sum+r;

num=num/10;

}while (num>0);

System.out.println("The sum of digits of " + n + " is " + sum);

}

}

1. **Switch case**

public class SwitchCase {

public static void main(String[] args) {

String day = "Tuesday";

switch (day){

case "Saturday":

case "Sunday":

System.out.println("Weekend");

break;

case "Monday":

case "Tuesday":

case "Wednesday":

case "Thursday":

case "Friday":

System.out.println("Weekday");

break;

default:

System.out.println("Invalid");

}

}

}

1. **Given the array consisting of 2n elements in the form [x1,x2,x3......xn,y1,y2,y3......yn]. Return the array in the form of [x1,y1,x2,y2...........xn,yn]**

import java.util.Scanner;

public class Array2n {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

int[] arr = new int[2\*n];

int[] res = new int[2\*n];

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

for(int i=0;i<2\*n;i++)

arr[i] = input.nextInt();

for(int i=0,j=n,k=0;i<n;i++,j++,k+=2) {

res[k] = arr[i];

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

}

for(int i=0;i<2\*n;i++)

System.out.print(res[i]);

}

}

1. **Difference between the sum and product of digits of a number**

import java.util.Scanner;

public class DiffSumProductOfDigits {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

int num = n;

int sum = 0;

int product = 1;

while(num>0){

int dig = num%10;

sum += dig;

product \*= dig;

num = num/10;

}

System.out.println("The sum of digits is "+ sum);

System.out.println("The product of digits is " + product);

System.out.print("The difference between sum and product of digits of " + n + " is ");

System.out.println(product-sum);

}

}

1. **Given an array of integers nums, return the number of good pairs. A pair (i,j) is said to be a good pair if nums[i]==nums[j] and i<j**

import java.util.Scanner;

public class GoodPairs {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

int[] nums = new int[n];

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

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

nums[i] = input.nextInt();

int count = 0;

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

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

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

System.out.println("(" + i + "," + j + ")");

count++;

}

}

}

System.out.println("No of good pairs is " + count);

}

}

1. **Given a valid(IPv4) IP address, return a defanged version of that IP address**

import java.util.Scanner;

public class ValidIp{

public static void main(String[] args)

{

System.out.print("Enter an Ip address : ");

Scanner input = new Scanner(System.in);

String address = input.next();

int len = address.length();

String defanged = "";

int i;

for(i=0;i<len;++i)

{

if(address.charAt(i)!='.')

defanged += address.charAt(i);

else

defanged += "[.]";

}

System.out.print("Defanged Ip Address is : " + defanged);

}

}

1. **Given an array of integers nums and an integer target, return indices of the two numbers such that they add upto target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.**

import java.util.Scanner;

public class TargetSum{

public static void main(String[] args)

{

int n;

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

Scanner input = new Scanner(System.in);

n = input.nextInt();

// declaring array

int[] arr = new int[n];

System.out.print("Enter array elements : ");

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

{

arr[i] = input.nextInt();

}

System.out.print("Enter target sum : ");

int targetSum = input.nextInt();

//actual logic to find pair of two indices such that integers at those indices

System.out.println();

System.out.println("Pair of indices that add upto targetSum are ");

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

{

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

{

if(arr[i] + arr[j] == targetSum)

System.out.println("(" + i + "," + j + ")");

}

}

System.out.println();

}

}

1. **Write a Java program to print all the twin primes below 1000. (A twin prime is a prime number that differs from another prime number by two. (3, 5), (5, 7), (11, 13),(17, 19), (29, 31), (41, 43), .821, 823), etc. .**

public class TwinPrime {

public static void main(String[] args) {

int n = 1000;

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

if(isPrime(i) && isPrime(i+2))

System.out.println("("+i+","+(i+2)+")");

}

}

static boolean isPrime(int n){

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

if(n%i==0)

return false;

}

return true;

}

}

1. **Write a Java program to implement matrix multiplication. (Take the input from keyboard).**

import java.util.Scanner;

public class MatrixMul {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[][] a = new int[5][5];

int[][] b = new int[5][5];

int[][] c = new int[5][5];

System.out.println("Matrix A");

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

int r1 = input.nextInt();

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

int c1 = input.nextInt();

System.out.println("Matrix A Elements");

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++) {

a[i][j] = input.nextInt();

}

}

System.out.println("Matrix B");

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

int r2 = input.nextInt();

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

int c2 = input.nextInt();

System.out.println("Matrix B Elements");

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

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

b[i][j] = input.nextInt();

}

}

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

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

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

c[i][j] += a[i][k] \* b[k][j];

}

}

}

System.out.println("The multiplied matrix is");

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

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

System.out.print(c[i][j]);

}

System.out.println();

}

}

}

1. **Write a Java program for sorting a given list of names in ascending order.**

import java.util.Arrays;

import java.util.Scanner;

public class SortNames {

public static void main(String[] args)

{

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

Scanner input = new Scanner(System.in);

int n = input.nextInt();

// create string array called names

String[] names = new String[n];

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

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

{

names[i] = input.next();

}

// inbuilt sort function

Arrays.sort(names);

// print output array

System.out.println();

System.out.println(

"The Names in alphabetical order are: ");

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

System.out.println(names[i]);

}

}

}

1. **The Fibonacci sequence is defined by the following rule. The first two values in the sequence are 1 and 1. Every subsequent value is the run of the two values preceding it. Write a Java program that uses both recursive and non-recursive functions to print the nth value in the Fibonacci sequence**

import java.util.Scanner;

public class Fibonacci{

static int recursiveFibonacci(int n)

{

if(n==1 || n==2)

return 1;

else

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

}

static int nonRecursiveFibonacci(int n)

{

if(n == 1 || n == 2)

return 1;

int a = 1 ;

int b = 1 ;

int i = 4 , r = a+b ;

while(i <= n){

a = b ;

b = r ;

r = a + b ;

i++ ;

}

return r;

}

public static void main(String[] args){

System.out.print("Enter nth term in Fibonacci Series (base : a=1, b=1) : ");

Scanner input = new Scanner(System.in);

int n = input.nextInt();

System.out.println(n + "th" + " term using recursion : " + recursiveFibonacci(n));

System.out.println(n + "th" + " term without recursion : " + nonRecursiveFibonacci(n));

}

}

1. **Write a Java program that prompts the user for an integer and then prints out all prime numbers up to that integer.**

import java.util.Scanner;

public class PrimeNumberUptoN {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

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

if(isPrime(i))

System.out.println(i);

}

}

static boolean isPrime(int n){

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

if(n%i==0)

return false;

}

return true;

}

}

1. **Write a java program that accepts an ‘n’ ordered square matrix elements into a single dimension array and print the elements of leading diagnol (top left to bottom right)**

import java.util.Scanner;

public class MatrixDiagnol {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int n = in.nextInt();

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

System.out.print("Enter matrix elements : ");

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

matrix[i] = in.nextInt();

}

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

System.out.println(matrix[n\*i+i]);

}

}

}

1. **Write a program that accepts an ‘’mxn’ double dimension array where m represents financial years and n represents IDs of items sold. Each element in the array represents no of items sold in a particular year. Identify year and ID of item which has more demand**

import java.util.Scanner;

public class FinancialYrAndItems {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.print("Enter m and n values : ");

int m = in.nextInt();

int n = in.nextInt();

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

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

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

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

}

}

int max=mat[0][0],maxItem=0,maxYear=0;

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

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

if(mat[i][j]>max){

max = mat[i][j];

maxItem = j;

maxYear = i;

}

}

}

System.out.println("The year with more demand is " + maxYear);

System.out.println("The item with more demand is " + maxItem);

}

}

1. **Write a program that accepts an n-ordered square matrix and calculate absolute difference between the sum of elements in their diagnol.**

import java.util.Scanner;

import java.lang.Math;

public class AbsoluteDiagonalDifference{

public static void main(String[] args){

Scanner input = new Scanner(System.in);

System.out.print("Enter dimension of square matrix: ");

int n = input.nextInt();

//square matrix creation

int[][] squareMatrix = new int[n][n];

// taking matrix as input from user

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

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

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

squareMatrix[i][j]=input.nextInt();

int sum1=0,sum2=0;

//leading diagonal sum

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

{

sum1 += squareMatrix[i][i];

}

//other diagonal

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

{

sum2 += squareMatrix[i][j];

}

System.out.println("Absolute Difference of sum of diagonals is " + Math.abs(sum1 - sum2));

}

}

1. **Examine datatypes and literals**

import java.util.\*;

public class DatatypeLiterals

{

public static void main(String[] args) {

int x = 0b1111;

double a = 0123.222;

int b = 0XFace;

System.out.println(x);

System.out.println(a);

System.out.println(b);

Scanner in = new Scanner(System.in);

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

long n = in.nextLong();

if(n>=-123 && n<=127)

{

System.out.println("Its a byte");

}

else if(n>=Short.MIN\_VALUE && n<= Short.MAX\_VALUE)

{

System.out.println("Its short");

}

else if(n>=Interger.MIN\_VALUE && n>= Integer.MAX\_VALUE)

{

System.out.println("Its int");

}

else if (n>= Long.MIN\_VALUE && n<=Long.MAX\_VALUE)

{

System.out.println("Its long");

}

}

}

1. **Move al the negative elements to one side of the array**

import java.util.Scanner;

public class NegativeMoveToEnd {

public static void main(String[] args) {

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

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int[] arr = new int[n];

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

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

arr[i] = sc.nextInt();

}

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

if(arr[i] < 0){

if(i!=j){

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

j++;

}

}

System.out.println("The new array is");

for (int i : arr) {

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

}

}

}

1. **Find the kth smallest element. Given an array and a number k where k is smaller than the size of array. The task is to find kth smallest in the given array. It is given that all array element are distinct.**

import java.util.Arrays;

import java.util.Scanner;

public class KthSmallestELement {

public static void main(String[] args) {

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

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int[] arr = new int[n];

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

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

arr[i] = sc.nextInt();

}

Arrays.sort(arr);

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

int k = sc.nextInt();

System.out.print("The kth smallest element: " + arr[k-1]);

}

}

1. **Given an integer, perform the following conditional actions.**

**If it is odd, print ‘Weird’**

**If it is even and in the range [6:10], print ‘Not weird’**

**If it is even and inclusive range of 15-20 print ‘Weird’**

**If it is even and greater than 30, print ‘Not Weird’**

import java.util.Scanner;

public class CheckWierd {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int n = in.nextInt();

if(n%2==1)

System.out.println("Weird");

else{

if(n>=6 && n<=10)

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

else if(n>=15 && n<=25)

System.out.println("Weird");

else if(n>=30)

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

}

}

}

1. **Print nth table**

import java.util.Scanner;

public class NthMultiplicationTable {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int n = in.nextInt();

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

System.out.println(n + " \* " + i + " = " + n\*i);

}

}

}

1. **Print the following sequence**

**(a+20b), (a+20 b+21 b),……..,(a+20 b+21 b+…..+2n-1 b)**

import java.util.Scanner;

public class Sequence {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.print("Enter a,b,n values : ");

int a = in.nextInt();

int b = in.nextInt();

int n = in.nextInt();

int ans = 0;

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

if(i==0)

ans += a + Math.pow(2,i) \* b;

else

ans += Math.pow(2,i) \* b;

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

}

}

}

1. **Write a Java program that checks whether a given string is a palindrome or not from command line. Ex: MALAYALAM is a palindrome.**

public class StringPalindrome {

public static void main(String[] args) {

String rev="";

for (int i = args[0].length()-1; i >= 0; i--) {

rev += args[0].charAt(i);

}

if(args[0].equals(rev))

System.out.println(args[0] + " is a palindrome");

else

System.out.println(args[0] + " is not a palindrome");

}

}

1. **Write a Java program that prints all real solutions to the quadratic equation ax2 + bx + c = 0. Read in a, b, c and use the quadratic formula. If the discriminant b2 - 4ac is negative, display a message stating that there are no real solutions**

import java.util.Scanner;

public class QuadraticRoots {

public static void main(String[] args) {

int a, b, c;

Scanner sc = new Scanner(System.in);

System.out.print("Enter the values of a, b and c: ");

a = sc.nextInt();

b = sc.nextInt();

c = sc.nextInt();

double discriminant = (b \* b) - 4 \* a \* c;

if(discriminant < 0){

System.out.println("There are no real roots. ");

}

else if(discriminant == 0){

double x = (-b)/2\*a;

System.out.println("Real and equal roots and the root is : " + x);

}

else{

double x = ( -b + Math.sqrt(discriminant) )/ 2\*a;

double x2 = ( -b - Math.sqrt(discriminant) )/ 2\*a;

System.out.println("Two roots are : "+ x + " " + x2);

}

}

}

1. **Given an array of strings, count the number of vowels and consonants in each string’**

import java.util.Scanner;

public class VowelsCount {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int n = in.nextInt();

String[] words = new String[n];

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

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

words[i] = in.next();

}

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

int vowel=0,consonant=0;

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

char s = words[i].toLowerCase().charAt(j);

if(s=='a' || s=='e' || s=='i' || s=='o' || s=='u')

vowel++;

else

consonant++;

}

System.out.println(words[i] + " Vowels: " + vowel + " consonants: " + consonant);

}

}

}

1. **Given two strings, check whether they are anagrams or not**

import java.util.Arrays;

import java.util.Scanner;

public class Anagrams {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter 2 strings");

String s1 = in.next().toLowerCase();

String s2 = in.next().toLowerCase();

char[] c1 = s1.toCharArray();

char[] c2 = s2.toCharArray();

Arrays.sort(c1);

Arrays.sort(c2);

System.out.println(Arrays.toString(c1));

System.out.println(Arrays.toString(c2));

if(Arrays.equals(c1,c2))

System.out.println(s1 + " and " + s2 + " are anagrams");

else

System.out.println(s1 + " and " + s2 + " are not anagrams");

}

}

1. **Var-args**

import java.util.Scanner;

public class VarArgs {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.print("Enter 2 numbers : ");

int a = in.nextInt();

int b = in.nextInt();

int sum2 = sum(a,b);

System.out.print("Enter 3 numbers : ");

int n1 = in.nextInt();

int n2 = in.nextInt();

int n3 = in.nextInt();

int sum3 = sum(n1,n2,n3);

System.out.println("The sum of " + a + " and " + b + " is " + sum2);

System.out.println("The sum of " + n1 + ", " + n2 + ", " + n3 + " is " + sum3);

}

static int sum(int... a) {

int sum = 0;

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

sum += a[i];

}

return sum;

}

}

1. **Area calculation using function overloading**

import java.util.Scanner;

public class AreaCalculation {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

double r = in.nextDouble();

double areaCircle = area(r);

System.out.print("Enter length and breadth of rectangle : ");

int l = in.nextInt();

int b = in.nextInt();

int areaRectangle = area(l,b);

System.out.print("Enter side of square : ");

int s = in.nextInt();

int areaSquare = area(s);

System.out.println("The area of circle is " + areaCircle);

System.out.println("The area of rectangle is " + areaRectangle);

System.out.println("The area of Square is " + areaSquare);

}

static double area(double r) {

return 3.14\*r\*r;

}

static int area(int l,int b) {

return l\*b;

}

static int area(int s) {

return s\*s;

}

}

1. **Create a class and an object of a class.**

**Create multiple objects of a class**

**Accessing class attributes**

**Modify attributes**

**Create a class method**

**Method with parameters**

**Access class methods with an object**

import java.util.Scanner;

public class ClassObjects {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

Vehicle car = new Vehicle();

System.out.println("Enter car details");

car.setName(in.next());

car.setColor(in.next());

car.setNoOfWheels(in.nextInt());

String name = car.getName();

String color = car.getColor();

int noOfWheels = car.getNoOfWheels();

System.out.println("Name: " + name + " Color: " + color + " No of wheels: " + noOfWheels);

System.out.println("Enter bike details");

Vehicle bike = new Vehicle();

bike.setName(in.next());

bike.setColor(in.next());

bike.setNoOfWheels(in.nextInt());

name = bike.getName();

color = bike.getColor();

noOfWheels = bike.getNoOfWheels();

System.out.println("Name: " + name + " Color: " + color + " No of wheels: " + noOfWheels);

System.out.println();

// accessing car name

System.out.println("The name of car is " + car.name);

System.out.println();

// modifying color attribute of car object

car.color = "Blue";

System.out.println("The color of car is " + car.color);

System.out.println();

// class method

Vehicle.className();

}

}

class Vehicle {

String name,color;

int noOfWheels;

void setName(String name) {

this.name = name;

}

void setColor(String color) {

this.color = color;

}

void setNoOfWheels(int noOfWheels) {

this.noOfWheels = noOfWheels;

}

String getName() {

return name;

}

String getColor() {

return color;

}

int getNoOfWheels() {

return noOfWheels;

}

static void className() {

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

}

}

1. **Write a program to create a room class, the attributes of this class are roomno, roomtype, roomarea and AC machine. In this class the member functions are setdata and displaydata.**

import java.util.Scanner;

public class RoomClass {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

Room roomObj = new Room();

System.out.print("Enter room no : ");

int roomNo = in.nextInt();

System.out.print("Enter room type : ");

String roomType = in.next();

System.out.print("Enter room area : ");

double roomArea = in.nextDouble();

System.out.print("Enter AC machine : ");

String acMachine = in.next();

roomObj.setData(roomNo,roomType,roomArea,acMachine);

roomObj.displayData();

}

}

class Room {

int roomNo;

String roomType, acMachine;

double roomArea;

void setData(int roomNo,String roomType,double roomArea,String acMachine) {

this.roomNo = roomNo;

this.roomType = roomType;

this.roomArea = roomArea;

this.acMachine = acMachine;

}

void displayData() {

System.out.println("Room No : " + roomNo);

System.out.println("Room Type : " + roomType);

System.out.println("Room Area : " + roomArea);

System.out.println("AC Machine : " + acMachine);

}

}

1. **Write a program create a class 'simpleobject'. Using constructor display the message**

public class Constructor {

public static void main(String[] args) {

SimpleObject obj = new SimpleObject();

}

}

class SimpleObject {

SimpleObject() {

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

}

}

1. **Write a program to give the example for 'this' operator. Also use the 'this' keyword as return statement**

public class ThisKeyword {

public static void main(String[] args) {

Student s1 = new Student("210","Irfan","CSE",19);

Student s2 = new Student();

s2 = s1.setValue();

s2.display();

}

}

class Student {

String rollNo,studentName,branch;

int age;

Student() {

System.out.println("New Student created");

}

Student(String rollNo, String studentName, String branch, int age) {

this.rollNo = rollNo;

this.studentName = studentName;

this.branch = branch;

this.age = age;

}

Student setValue() {

return this;

}

void display() {

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

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

System.out.println("Branch : " + branch);

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

}

}

1. **Static variables, methods and blocks**

public class Static {

public static void main(String[] args) {

Car.displayDetails();

}

}

class Car {

static int noOfWheels;

static String fuelType;

static {

noOfWheels = 4;

fuelType = "Petrol";

}

static void displayDetails() {

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

System.out.println("Fuel Type : " + fuelType);

}

}

1. **Instance variables, methods and blocks**

public class Instance {

public static void main(String[] args) {

Bike b1 = new Bike();

b1.displayDetails();

}

}

class Bike {

String name,color;

{

name = "bullet";

color = "black";

}

void displayDetails() {

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

}

}

1. **Function Overloading**

public class FunctionOverloading {

public static void main(String[] args) {

main(1.2f,1.3f,2.2f);

main(1,2);

main("Irfan ","Teja");

}

public static void main(float a,float b,float c) {

System.out.println("Main float Sum is " + (a+b+c));

}

public static void main(int a,int b) {

System.out.println("Main int Sum is " + (a+b));

}

public static void main(String s1,String s2) {

System.out.println("Main String concatenation " + s1+s2);

}

}

1. **Constructor Overloading**

public class ConstructorOverloading {

public static void main(String[] args) {

Vehicle auto = new Vehicle();

Vehicle car = new Vehicle("Ferrari");

Vehicle bike = new Vehicle("KTM","blue");

}

}

class Vehicle {

String brand,color;

Vehicle() {

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

}

Vehicle(String brand) {

this.brand = brand;

System.out.println("Brand : " + brand);

}

Vehicle(String brand, String color) {

this.brand = brand;

this.color = color;

System.out.println("Brand : " + brand + " Color : " + color);

}

}

1. **Create class named as 'a' and create a sub class 'b' which is extends from class 'a'. And use these classes in 'Inherit' class**

public class Inheritance {

public static void main(String[] args) {

Inherit obj = new Inherit();

obj.objA.display();

obj.objB.display();

}

}

class Inherit {

Inherit() {

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

}

A objA = new A();

B objB = new B();

}

class A {

A() {

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

}

public void display() {

System.out.println("Class A is a super class");

}

}

class B extends A {

B() {

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

}

public void display() {

System.out.println("Class B is inherited from class A");

}

}

1. **Heirarchial Inheritance**

public class HeirarchialInheritance {

public static void main(String[] args) {

Boy b1 = new Boy();

Girl g1 = new Girl();

}

}

class Parent {

Parent() {

System.out.println("Parents love children");

}

}

class Boy extends Parent {

Boy() {

System.out.println("Mama's boy");

}

}

class Girl extends Parent {

Girl() {

System.out.println("Daddy's girl");

}

}

1. **Multilevel Inheritance**

public class MultiLevelInheritance {

public static void main(String[] args) {

Tiger t1 = new Tiger();

}

}

class Animal {

Animal() {

System.out.println("Animal class");

}

}

class WildAnimal extends Animal {

WildAnimal() {

System.out.println("Wild Animal class");

}

}

class Tiger extends WildAnimal {

Tiger() {

System.out.println("Tiger class");

}

}

1. **Create a class Box that uses a parameterised constructor to initialise 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**

public class BoxClass {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter width,height and depth of a box");

Box boxObj = new Box(in.nextInt(),in.nextInt(), in.nextInt());

int vol = boxObj.volume();

System.out.print("The volume of the box is " + vol );

}

}

class Box {

int width,height,depth;

Box(int width,int height,int depth) {

this.width = width;

this.height = height;

this.depth = depth;

}

int volume() {

return width\*height\*depth;

}

}

1. **Create a new class called Calculator with the following methods. A static method called powerInt(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 both the methods and test the functionality. Also count the number of objects**

import java.util.Scanner;

public class CalculatorClass {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.print("Enter 2 integers : ");

double res1 = Calculator.powerInt(in.nextInt(),in.nextInt());

System.out.println("The power of integers is " + res1);

System.out.print("Enter 2 doubles : ");

double res2 = Calculator.powerDouble(in.nextDouble(),in.nextDouble());

System.out.println("The power of doubles is " + res2);

}

}

class Calculator {

static double powerInt(int num1,int num2) {

return Math.pow(num1,num2);

}

static double powerDouble(double num1,double num2) {

return Math.pow(num1,num2);

}

}

1. **Final variable**

public class FinalVariables {

public static void main(String[] args) {

final int a = 10;

System.out.println("The value of a is " + a);

a = 100; // The final variable cannot be modified

}

}

1. **Final method**

public class FinalMethod {

public static void main(String[] args) {

Dog d1 = new Dog();

d1.eat();

d1.bark();

}

}

class Animals {

final void eat() {

System.out.println("eats food");

}

}

class Dogs extends Animals {

// Final methods cannot be overriden

void eat() {

System.out.println("eats pedigree");

}

void bark() {

System.out.println("Dog barks");

}

}

1. **Final class**

public class FinalClass {

public static void main(String[] args) {

Honda h = new Honda();

}

}

final class Cars {

Cars() {

System.out.println("Cars class");

}

}

class Honda extends Cars {

Honda() {

System.out.println("Honda class");

}

}

1. **Method overriding**

public class MethodOverriding {

public static void main(String[] args) {

SBI s = new SBI();

ICICI i = new ICICI();

AXIS a = new AXIS();

System.out.println("SBI Rate of interest is " + s.getRateOfInterest());

System.out.println("ICICI Rate of interest is " + i.getRateOfInterest());

System.out.println("AXIS Rate of interest is " + a.getRateOfInterest());

}

}

class Bank {

int getRateOfInterest() {

return 0;

}

}

class SBI extends Bank {

int getRateOfInterest() {

return 8;

}

}

class ICICI extends Bank {

int getRateOfInterest() {

return 7;

}

}

class AXIS extends Bank {

int getRateOfInterest() {

return 9;

}

}

1. **Dynamic method dispatch**

public class DynamicMethodDispatch {

public static void main(String[] args) {

Bank b1 = new SBI();

Bank b2 = new ICICI();

Bank b3 = new AXIS();

System.out.println("SBI Rate of interest is " + b1.getRateOfInterest());

System.out.println("ICICI Rate of interest is " + b2.getRateOfInterest());

System.out.println("AXIS Rate of interest is " + b3.getRateOfInterest());

}

}

class Bank {

int getRateOfInterest() {

return 0;

}

}

class SBI extends Bank {

int getRateOfInterest() {

return 8;

}

}

class ICICI extends Bank {

int getRateOfInterest() {

return 7;

}

}

class AXIS extends Bank {

int getRateOfInterest() {

return 9;

}

}

1. **Super variables**

public class SuperMembers {

public static void main(String[] args) {

Car obj = new Car();

obj.showColor();

}

}

class Vehicle {

String color = "white";

}

class Car extends Vehicle {

String color = "black";

void showColor() {

System.out.println("Vehicle color is " + super.color);

System.out.println("Car color is " + color);

}

}

1. **Super method**

public class SuperMethod {

public static void main(String[] args) {

Dog d1 = new Dog();

d1.displayMethods();

}

}

class Animal {

void eat() {

System.out.println("Animal eats food");

}

}

class Dog extends Animal {

void eat() {

System.out.println("Dog eats pedigree");

}

void bark(){

System.out.println("Dog barks");

}

void displayMethods() {

super.eat();

eat();

bark();

}

}

1. **Super constructor**

import java.util.Arrays;

public class SuperConstructor {

public static void main(String[] args) {

Square s1 = new Square();

}

}

class Shape {

Shape() {

System.out.println("A shape is created");

}

Shape(int... a) {

System.out.println("The dimensions of shape are " + Arrays.toString(a));

}

}

class Square extends Shape {

Square() {

super(1,2,3);

System.out.println("A square is created");

}

}

1. **Area of triangle using super**

public class TriangleArea {

public static void main(String[] args) {

RightAngledTriangle obj = new RightAngledTriangle(3.0,4.0,5.0);

obj.displayArea();

}

}

class Triangle {

double a,b,c,area;

Triangle(double a,double b,double c) {

this.a = a;

this.b = b;

this.c = c;

}

void calcArea() {

double s = (a+b+c)/2;

area = Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

System.out.println("Area of triangle is " + area);

}

}

class RightAngledTriangle extends Triangle {

double b,h;

RightAngledTriangle(double a,double b,double c) {

super(a,b,c);

this.b = a;

this.h = b;

}

void calcArea() {

area = 0.5\*b\*h;

System.out.println("Area of right angled triangle is " + area);

}

void displayArea() {

super.calcArea();

calcArea();

}

}

1. **Create class point with following instance variables and methods. Instance variable : private int x,y Constructors : public Point(), Point(int x,int y). Methods : public void setX(int x), setY(int y), setXY(int x,int y) and also find the distance between the points.**

public class PointClass {

public static void main(String[] args) {

Point p = new Point();

p.distance();

}

}

class Point {

private int x,y;

public Point() {

}

Point(int x,int y) {

this.x = x;

this.y = y;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public void setXY(int x,int y) {

this.x = x;

this.y = y;

}

void distance() {

Point p1 = new Point(2,3);

Point p2 = new Point();

p2.setX(2);

p2.setY(0);

Point p3 = new Point();

p3.setXY(1,1);

double d1 = Math.sqrt(((p1.x-p2.x)\*(p1.x-p2.x)) + ((p1.y-p2.y)\*(p1.y-p2.y)));

double d2 = Math.sqrt(((p2.x-p3.x)\*(p2.x-p3.x)) + ((p2.y-p3.y)\*(p2.y-p3.y)));

double d3 = Math.sqrt(((p3.x-p1.x)\*(p3.x-p1.x)) + ((p3.y-p1.y)\*(p3.y-p1.y)));

display(p1,p2,p3,d1,d2,d3);

}

void display(Point p1,Point p2,Point p3,double d1,double d2,double d3){

System.out.println("The distance between (" + p1.x + "," + p1.y + ")" + " and (" + p2.x + "," + p2.y + ")" + " is " + d1);

System.out.println("The distance between (" + p2.x + "," + p2.y + ")" + " and (" + p3.x + "," + p3.y + ")" + " is " + d2);

System.out.println("The distance between (" + p1.x + "," + p1.y + ")" + " and (" + p3.x + "," + p3.y + ")" + " is " + d3);

}

}

1. **Create a class Number with only one private instance variable as a double primitive type. To include the following methods (include respective constructors) isZero(), isPositive(), isNegative(), isOdd(), isEven(), isPrime(), isArmstrong() the above methods return boolean primitive type. getFactorial, getSqrt(), getSqr(), sumDigits(), getReverse() the above methods return double primitive type. void listFactor(), void dispBinary()**

public class NumberClass {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int num = in.nextInt();

Number numObj = new Number(num);

System.out.println(num + " is zero? : " + numObj.isZero());

System.out.println(num + " is positive? : " + numObj.isPositive());

System.out.println(num + " is negative? : " + numObj.isNegative());

System.out.println(num + " is even? : " + numObj.isEven());

System.out.println(num + " is odd? : " + numObj.isOdd());

System.out.println(num + " is prime? : " + numObj.isPrime());

System.out.println(num + " is armstrong? : " + numObj.isArmstrong());

System.out.println(num + "! = " + numObj.factorial());

System.out.println("Square root of " + num +" is " + numObj.getSqrt());

System.out.println("Square of " + num +" is " + numObj.getSqr());

System.out.println("Digit sum of " + num + " is " + numObj.sumDigit());

System.out.println("Reverse of " + num + " is " + numObj.getReverse());

numObj.listFactors();

numObj.dispBinary();

}

}

class Number {

private int num;

Number(int num) {

this.num = num;

}

boolean isZero() {

return num==0;

}

boolean isPositive() {

return num>0;

}

boolean isNegative() {

return num<0;

}

boolean isEven() {

return num%2==0;

}

boolean isOdd() {

return num%2==1;

}

boolean isPrime() {

if(num<2)

return false;

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

if(num%i==0)

return false;

}

return true;

}

boolean isArmstrong() {

int n = num;

int s = 0;

while(n>0) {

int r = n%10;

s += r\*r\*r;

n = n/10;

}

if(s==num)

return true;

else

return false;

}

int factorial() {

int fact = 1;

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

fact \*= i;

}

return fact;

}

double getSqrt() {

return Math.sqrt(num);

}

int getSqr() {

return num\*num;

}

int sumDigit() {

int n = num;

int sum = 0;

while(n>0) {

int r = n%10;

sum += r;

n = n/10;

}

return sum;

}

int getReverse() {

int n = num;

int rev = 0;

while(n>0) {

int r=n%10;

rev = rev\*10 + r;

n = n/10;

}

return rev;

}

void listFactors() {

if(num<1)

System.out.println(num + " doesn't have factors");

else {

System.out.print("The factors of " + num + " are ");

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

if (num % i == 0)

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

}

System.out.println();

}

}

void dispBinary() {

int n = num;

int s = 0;

int i = 0;

while(n>0) {

int r= n%2;

s = s + r\*(int)Math.pow(10,i);

n = n/2;

i++;

}

System.out.println("The binary representation of " + num + " is " + s);

}

}

1. **Nested classes**

public class NestedClass {

public static void main(String[] args) {

OuterClass.StaticNestedClass nestedObj = new OuterClass.StaticNestedClass();

nestedObj.display();

}

}

class OuterClass {

static int outer\_x = 10;

int outer\_y = 20;

private static int outer\_private = 30;

static class StaticNestedClass {

void display() {

System.out.println("Outer\_x = " + outer\_x);

System.out.println("Outer\_private = " + outer\_private);

// System.out.println("Outer\_y = " + outer\_y); cannot access non-static members of outer class

}

}

}

1. **Inner classes**

public class InnerClass {

public static void main(String[] args) {

new Outer().new Inner().method1(1000,2000);

}

}

class Outer{

private int a = 10;

private int b = 20;

class Inner{

int a = 100;

int b = 200;

void method1(int a,int b){

System.out.println("local : a is " + a + " and b is " + b);

System.out.println("Inner class : a is " + this.a + " and b is " + this.b);

System.out.println("Outer class : a is " + Outer.this.a + " and b is " + Outer.this.b);

}

}

}

1. **Local Inner classes**

public class LocalInnerClass {

public static void main(String[] args) {

Outer o = new Outer();

o.method1();

}

}

class Outer {

private int a = 100;

void method1(){

class Inner {

void method2() {

System.out.println("Inner method");

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

}

}

Inner i = new Inner();

i.method2();

}

}

1. **Anonymous inner classes**

public class AnonymousInnerClass {

A a = new A() {

void m1() {

System.out.println("m1 method");

}

void m2() {

System.out.println("m2 method");

}

};

public static void main(String[] args) {

AnonymousInnerClass t = new AnonymousInnerClass();

t.a.m1();

t.a.m2();

}

}

class A {

void m1() {}

void m2() {}

}

1. **Write a program to create a package named mypack and import it in circle class**

package Mypack;

public class Area{

public static double areaCalc(double r){

return 3.14\*r\*r;

}

}

import java.util.\*;

import Mypack.Area;

public class Circle {

public static void main(String[] args) {

double r = 5;

double area = Area.areaCalc(r);

System.out.println("Area of circle is " + area);

}

}

1. **Write a program to create a package named Pi and implement this package in Ex1 class**

package Pi;

public class PiClass{

public static double Value(){

return Math.pi;

}

}

import java.util.\*;

import Pi.PiClass;

public class Ex1 {

public static void main(String[] args) {

System.out.println("Enter radius of a sphere : ");

Scanner in = new Scanner(System.in);

double r = in.nextDouble();

double vol = (4/3)Piclass.value()(r\*r\*r);

System.out.println("Volume of sphere is "+vol);

}

}

1. **Define an interface using Java that contains a method to calculate the perimeter of an object. Define two classes Circle and Rectangle with suitable fields and methods. Implement the interface ‘Perimeter’ in these classes. Write the appropriate main() method to create object of each class and test all the methods­­­­­**

interface Perimeter {

public double getPerimeter();

}

public class PerimeterCalc {

public static void main(String[] args) {

Rectangle r1 = new Rectangle(4,2);

System.out.println("Rectangle Perimeter : " + r1.getPerimeter());

Circle c1 = new Circle(5);

System.out.println("Circle Perimeter : " + c1.getPerimeter());

}

}

class Rectangle implements Perimeter {

int l,b;

Rectangle(int l,int b) {

this.l = l;

this.b = b;

}

public double getPerimeter() {

return 2\*(l+b);

}

}

class Circle implements Perimeter {

int r;

Circle(int r) {

this.r = r;

}

public double getPerimeter() {

return 2\*3.14\*r;

}

}

1. **Exploration of String class with 10 methods**

import java.util.Scanner;

public class StringClass {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

String s = in.nextLine();

System.out.println("The string is " + s);

System.out.println("The length is " + s.length());

int i=7;

System.out.println("The character at " + i + "th position is " + s.charAt(i));

String t = "Java";

System.out.println("Checking if string is " + t + " : " + s.equals(t));

System.out.println("Checking if string contains " + t + " : " + s.contains(t));

System.out.println("Replacing ' ' with '#' : " + s.replace(' ','#'));

System.out.println("Substring from indices " + i + " to " + (i+4) + " is " + s.substring(i,i+4));

t = " and Python";

System.out.println("Concatenation : " + s.concat(t));

System.out.println("Index of J is " + s.indexOf('J'));

System.out.println("join method : " + s.join(",","Python","Java","C++"));

s = " Java";

System.out.println("strip method of " + s + " : " + s.strip());

}

}

1. **Create an interface named Polygon. It has default method getSides() and an abstract method getArea(). Create two classes Rectangle and Square that implements Polygon. The Rectangle class provides the implementation of the getArea() method and overrides the getSides() method. The square class only provides the implementation of the getArea() method. Observe while calling the getSides() method using the Rectangle object, the overridden method is called. However, in the case of the Square object, the default method is called**

interface Polygon {

default void getSides(){

System.out.println("getSides called");

}

abstract int getArea();

}

class Rectangle implements Polygon {

int l,b;

Rectangle(int l,int b) {

this.l = l;

this.b = b;

}

public void getSides() {

System.out.println("Length : " + l);

System.out.println("Breadth : " + b);

}

public int getArea() {

return l\*b;

}

}

class Square implements Polygon {

int s;

Square(int s) {

this.s = s;

}

public int getArea() {

return s\*s;

}

}

public class PolygonClass {

public static void main(String[] args) {

Rectangle recObj = new Rectangle(2,3);

Square sObj = new Square(5);

recObj.getSides();

System.out.println("Rectangle area is " + recObj.getArea());

System.out.println("Square area is " + sObj.getArea());

}

}

1. **Multiple Inheritance**

interface Printable {

public void print();

}

interface Showable {

public void show();

}

class A implements Printable,Showable {

public void print() {

System.out.println("Printing...");

}

public void show() {

System.out.println("Showing...");

}

}

public class MultipleInheritance {

public static void main(String[] args) {

A obj = new A();

obj.print();

obj.show();

}

}

1. **In the exercise template you'll find interface TacoBox ready to your use. It has the following methods**

**the method int tacosRemaining() return the number of tacos remaining in the box**

**the method void eat() reduces the number of tacos remaining by one. The number of tacos remaining can't be negative.**

**Implement the class TripleTacoBox, that implements the TacoBox interface. TripleTacoBox has a constructor with no parameters**

**TripleTacoBox has an object variable tacos which is initialised at 3 when the constructor is called**

interface TacoBox {

public int tacosRemaining();

public void eat();

}

class TripleTacoBox implements TacoBox {

int tacos;

TripleTacoBox() {

tacos = 3;

}

public int tacosRemaining() {

return tacos;

}

public void eat() {

tacos--;

}

}

public class TacBoxInterface {

public static void main(String[] args) {

TripleTacoBox tacos = new TripleTacoBox();

for (int i = 0; i <= 3; i++) {

tacos.eat();

if(tacos.tacosRemaining()<0) {

System.out.println("Sorry! No tacos");

}

else {

System.out.println("Tacos Remaining : " + tacos.tacosRemaining());

}

}

}

}

1. **Can inheritance be applied between interfaces. Illustrate with an example**

interface Printable {

public void print();

}

interface Showable {

public void show();

}

class Works implements Showable {

public void print() {

System.out.println("Printing ...");

}

public void show() {

System.out.println("Showing ...");

}

}

public class InterfaceInheritance {

public static void main(String[] args) {

Works w = new Works();

w.print();

w.show();

}

}

1. **1. Create an abstract class pen with methods, write() and refill() as abstract methods**

**2. Use the pen class Q1 to create a concrete class fountain pen with additional method changeNib()**

public class PenClass {

public static void main(String[] args) {

FountainPen p1 = new FountainPen();

p1.write();

p1.refill();

p1.color();

}

}

abstract class Pen {

abstract void write();

abstract void refill();

}

class FountainPen extends Pen {

void write() {

System.out.println("Fountain Pen writing ...");

}

void refill() {

System.out.println("Refilling ...");

}

void color() {

System.out.println("Blue color");

}

}

1. **Define a class called Student with the name and roll no.**

**Create five student objects as an array and sort them as per their roll no.**

import java.util.Scanner;

public class StudentClass {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

Student sObj[] = new Student[5];

System.out.println("Enter roll no and name");

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

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

int rollNo = in.nextInt();

String name = in.next();

sObj[i] = new Student(name,rollNo);

}

}

}

class Student {

String name;

int rollNo;

Student(String name, int rollNo) {

this.name = name;

this.rollNo = rollNo;

}

}

1. **Garbage Collection**

class Test extends Object{

public void finalize(){

System.out.println("Object destroyed");

}

}

public class GarbageCollection {

public static void main(String[] args) {

Test t1 = new Test();

Test t2 = new Test();

System.out.println(t1.toString());

System.out.println(t2.toString());

t1=null;

t2=null;

System.gc();

}

}

1. **Try-catch block**

import java.util.Scanner;

public class TryCatchDemo {

public static void main(String[] args) {

int n1,n2;

try{

n1=0;

n2=100/n1;

System.out.println("End of try block");

}

catch(ArithmeticException e){

System.out.println("Divide by 0 error");

}

catch(Exception e){

System.out.println("Exception");

}

System.out.println("Out of try-catch block");

}

}

1. **Multiple catch block**

public class MultipleCatchDemo {

public static void main(String[] args) {

try {

int[] a = new int[5];

a[10] = 60 / 0;

System.out.println("End of try block");

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array Index Out of Bounds Exception occured");

} catch (ArithmeticException e) {

System.out.println("Arithematic exception occured");

} catch (Exception e) {

System.out.println("Exception occured");

}

System.out.println("End of try-catch block");

}

}

1. **Checked exception**

public class CheckedException {

void studentDetails() throws InterruptedException{

System.out.println("Sleeping");

Thread.sleep(1000);

System.out.println("Do not disturb");

}

void hod() throws InterruptedException {

System.out.println("in hod begin");

studentDetails();

System.out.println("hod end");

}

void principal(){

System.out.println("principal start");

try{

hod();

}

catch(InterruptedException e){

System.out.println("InterruptedException occured");

}

System.out.println("principal end");

}

void officeBoy(){

principal();

}

public static void main(String[] args) {

CheckedException ce = new CheckedException();

ce.officeBoy();

}

}

1. **User Defined exception**

import java.util.Scanner;

class UserDefined extends Exception{

UserDefined(String s){

super(s);

}

}

public class UserDefinedException {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

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

int age = in.nextInt();

try{

if(age<18){

throw new UserDefined("Age must be greater than 18");

}

}

catch(UserDefined e){

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

}

}

}

1. **Try catch with finally**

public class TestBlockFinally {

public static void main(String[] args) {

try{

System.out.println("Inside try block");

int x=20/0;

System.out.println("Ënd of try");

}

catch (ArithmeticException e){

System.out.println("Exception Handled");

}

finally{

System.out.println("finally block is always executed");

}

System.out.println("rest of code...");

}

}

1. **Nested try catch**

public class NestedTryBlock {

public static void main(String[] args) {

try {

try{

try{

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

System.out.println(arr[10]);

}

catch (ArithmeticException e){

System.out.println("Arithematic exception");

System.out.println("inner try block 2");

}

}

catch (ArithmeticException e){

System.out.println("Arithematic exception");

System.out.println("Inner try block");

}

}

catch (ArrayIndexOutOfBoundsException e){

System.out.println(e);

System.out.println("outer try block");

}

catch (Exception e){

System.out.println("Exception");

System.out.println("Handled in main try block");

}

}

}

1. **Insufficient funds**

import java.util.Scanner;

class UserDefinedException extends Exception{

UserDefinedException(String s){

super(s);

}

}

public class InsufficientFunds {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int bal = 10000;

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

int amt = in.nextInt();

try {

if (amt > bal) {

throw new UserDefinedException("Insufficient funds");

} else {

System.out.println("Balance is " + (bal - amt));

}

}

catch(UserDefinedException e){

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

}

}

}

1. **Thread creation with Thread class**

class MyThread extends Thread{

public void run() {

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

}

}

public class ThreadDemo {

public static void main(String[] args) {

MyThread t = new MyThread();

t.start();

}

}

1. **Thread creation using Runnable interface**

class MyThread2 implements Runnable{

public void run(){

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

}

}

public class ThreadDemo2 {

public static void main(String[] args) {

MyThread2 t = new MyThread2();

Thread th = new Thread(t);

th.start();

}

}

1. **Multithreading**

class Thread1 implements Runnable{

public void run(){

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

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

}

}

}

public class MultiThreading {

public static void main(String[] args) {

Thread1 obj1 = new Thread1();

Thread1 obj2 = new Thread1();

Thread t1 = new Thread(obj1,"Thread1");

Thread t2 = new Thread(obj2,"Thread2");

t1.start();

t2.start();

}

}

1. **Thread names - getName(),setName()**

class MyThread3 extends Thread{

public void run(){

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

System.out.println(Thread.currentThread().getName());

}

}

}

public class ThreadNames {

public static void main(String[] args) {

MyThread3 t1 = new MyThread3();

MyThread3 t2 = new MyThread3();

t1.setName("Thread 1");

t2.setName("Thread 2");

t1.start();

t2.start();

}

}

1. **Thread priorities**

class MyThread1 extends Thread{

}

public class ThreadPriority {

public static void main(String[] args) {

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

Thread.currentThread().setPriority(Thread.MAX\_PRIORITY);

MyThread1 t = new MyThread1();

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

}

}

1. **Thread Interruptions – yield()**

class MyThread4 extends Thread{

public void run(){

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

Thread.yield();

System.out.println("child thread");

}

}

}

public class ThreadYieldDemo {

public static void main(String[] args) {

MyThread4 t = new MyThread4();

t.setPriority(Thread.MAX\_PRIORITY);

t.start();

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

System.out.println("main thread");

}

}

}

1. **Thread Interruptions – join()**

class MyThread5 extends Thread{

public void run() {

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

System.out.println("child thread");

try{

Thread.sleep(1000);

}

catch (Exception e){

System.out.println("exception occurred");

}

}

}

}

public class ThreadJoinDemo2 {

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

MyThread5 t = new MyThread5();

t.start();

t.join();

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

System.out.println("main thread");

}

}

}

1. **Thread Interruptions – sleep()**

class MyThread2 extends Thread{

public void run(){

try{

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

System.out.println("i am lazy thread");

Thread.sleep(1000);

}

}

catch(InterruptedException e){

System.out.println("i got interrupted");

}

}

}

public class ThreadSleep {

public static void main(String[] args) {

MyThread2 t = new MyThread2();

t.start();

t.interrupt();

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

System.out.println("main thread");

}

}

}

1. **Synchronized**

class Message{

public synchronized void greet(String s){

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

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

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println("interrupted");

}

System.out.println(s);

}

}

}

class MyThread3 extends Thread{

Message m;

String name;

MyThread3(Message m,String name) {

this.m = m;

this.name = name;

}

public void run(){

m.greet(name);

}

}

public class Synchronization {

public static void main(String[] args) {

Message m = new Message();

MyThread3 mt1 = new MyThread3(m,"teja");

MyThread3 mt2 = new MyThread3(m,"Irfan");

mt1.start();

mt2.start();

}

}

1. **Inter thread communication**

class ThreadA extends Thread{

int total = 0;

public void run(){

synchronized (this){

System.out.println("child thread starts calculation");

total = 100;

System.out.println("child thread gives notify() call");

this.notify();

}

}

}

public class InterThreadCommunication {

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

ThreadA t = new ThreadA();

t.start();

synchronized (t){

System.out.println("main thread calling wait()");

t.wait();

System.out.println("main thread gets notify() call");

System.out.println(t.total);

}

}

}

1. **Producer Consumer problem**

class p extends Thread{

StringBuffer b ;

p(){

b= new StringBuffer(5);

}

public void run(){

synchronized (b){

try{

for (int i = 0; i < b.capacity(); i++) {

b.append(i);

System.out.println("Produced " + i);

}

}

catch(Exception e){

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

}

System.out.println("BUFFER is full");

b.notify();

}

}

}

class c extends Thread{

p obj;

c(p temp){

obj = temp;

}

public void run(){

synchronized (obj.b){

System.out.println("Consuming data");

try{

obj.b.wait();

}

catch(Exception e){

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

}

for (int i = 0; i < obj.b.capacity(); i++) {

System.out.println("Consumed "+ obj.b.charAt(i));

}

System.out.println("Buffer is empty");

}

}

}

public class ProducerConsumer extends Thread{

public static void main(String[] args) {

p obj = new p();

c obj2 = new c(obj);

obj2.start();

obj.start();

}

}

1. **Deadlock**

public class DeadLockDemo {

public static void main(String[] args) {

final String resource1 = "Resource 1";

final String resource2 = "Resource 2";

Thread t1 = new Thread() {

public void run() {

synchronized (resource1) {

System.out.println("Thread1 locked resource1");

try {

Thread.sleep(100);

} catch (InterruptedException e) {

System.out.println("exception occurred");

}

synchronized (resource2) {

System.out.println("Thread1 locked resource2");

}

}

}

};

Thread t2 = new Thread(){

public void run() {

synchronized (resource2) {

System.out.println("Thread2 locked resource2");

try {

Thread.sleep(100);

} catch (InterruptedException e) {

System.out.println("exception occurred");

}

synchronized (resource1) {

System.out.println("Thread2 locked resource1");

}

}

}

};

t1.start();

t2.start();

}

}

1. **Collections – ArrayList**

import java.util.ArrayList;

public class ArrayListDemo {

public static void main(String[] args) {

ArrayList al = new ArrayList();

al.add(10);

al.add("rat");

al.add("teja");

al.add('a');

al.add(null);

al.add(10);

System.out.println("ArrayList data : " + al);

System.out.println("ArrayList size : " + al.size());

al.add(1,"Hello");

System.out.println("after adding objects ArrayList size : " + al.size());

System.out.println("ArrayList data : " + al);

al.remove(1);

al.remove("hello");

System.out.println("after removing ArrayList size : " + al.size());

System.out.println("ArrayList data : " + al);

System.out.println("ArrayList Empty ? : " + al.isEmpty());

al.clear();

System.out.println("ArrayList Empty ? : " + al.isEmpty());

}

}

1. **LinkedList**

import java.util.LinkedList;

public class LinkedListDemo {

public static void main(String[] args) {

LinkedList l = new LinkedList();

System.out.println(l);

l.add(123);

l.add((Integer) 123);

l.addFirst(1);

l.addLast(10);

l.add(123);

System.out.println(l);

System.out.println("has 123? " + l.contains(123));

System.out.println("Get First: " + l.getFirst());

System.out.println("last removed " + l.removeLast());

System.out.println(l.toString());

l.removeFirst();

l.removeFirstOccurrence("removing last occurence of 123" + 123);

System.out.println(l);

System.out.println(" Printing method l.toArray()" + l.toArray());

}

}

1. **Vector**

import java.util.Enumeration;

import java.util.Vector;

public class VectorDemo {

public static void main(String[] args) {

Vector v = new Vector();

System.out.println("Printing " + v);

v.add(123);

v.addElement(123);

v.addElement(143);

v.addElement(3);

v.addElement(343);

v.addElement(453);

v.addElement(53);

v.addElement(1);

System.out.println(v);

v.remove(0);

System.out.println(v);

System.out.println("Printing"+v);

System.out.println("capacity: " + v.capacity());

System.out.println("firstElement" + v.firstElement());

System.out.println("isEmpty" + v.isEmpty());

System.out.println("lastElement" + v.lastElement());

for (Object o : v) {

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

}

}

}

1. **HashSet**

import java.util.\*;

public class HashSetExample {

public static void main(String[] args) {

HashSet<String> set = new HashSet();

set.add("32");

set.add("23");

set.add("43"); set.add("546"); set.add("6145");

System.out.println("Removing the object 23" + set.remove("23"));

System.out.println("Is empty: " + set.isEmpty());

System.out.println( "Printing to array" + set.toArray());

System.out.println("Size: " + set.size());

Iterator i = set.iterator();

while(i.hasNext()){

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

}

}

}

1. **LinkedHashSet**

import java.util.ArrayList;

import java.util.LinkedHashSet;

import java.util.\*;

public class LinkedHashSetExample {

public static void main(String[] args) {

LinkedHashSet h = new LinkedHashSet();

h.add(12);

h.add(342);

h.add(24);

h.add(132);

System.out.println("Removing obj 342"+h.remove(342));

System.out.println("Size:" + h.size());

ArrayList a = new ArrayList();

a.add(23); a.add(233); a.add(54);

System.out.println("" + h.addAll(a));

Iterator i = h.iterator();

while(i.hasNext()){

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

}

}

}

1. **TreeSet**

import java.util.TreeSet;

public class TreeSetDemo {

public static void main(String[] args) {

TreeSet ts = new TreeSet();

ts.add(12); ts.add(452);ts.add(34543);ts.add(635); ts.add(789654321);

System.out.println("Ceiling: " + ts.ceiling(12));

System.out.println("Floor of 123.123: " + ts.floor(123));

System.out.println("Contains 12?: " + ts.contains(12));

System.out.println("First element: " + ts.first());

System.out.println("Lower of" + ts.lower(12));

System.out.println(ts);

}

}

1. **HashMap**

package Collections;

import javax.swing.text.html.HTMLDocument;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;

public class HashMapDemo {

public static void main(String[] args) {

HashMap<Integer,Integer> hm = new HashMap<>();

System.out.print("Enter the no of elem in hashmap:");

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

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

hm.put(i,sc.nextInt());

}

System.out.println("Whether contains 5" + hm.containsValue(5));

System.out.println("Whether contains 4 as key" + hm.containsKey(4));

System.out.println("replacing 23 value for 1 key" + hm.replace(1,23));

System.out.println();

System.out.println("Removing object with key 2"+ hm.remove(2));

System.out.println("Size of the hashmap:" + hm.size());

System.out.println("Printing the elements/values");

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

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

}

}

}

1. **HashTable**

package Collections;

import java.util.Hashtable;

import java.util.Iterator;

public class HashTableDemo {

public static void main(String[] args) {

Hashtable<Integer, Integer> ht = new Hashtable();

ht.put(5, 423);

ht.put(4, 12);

ht.put(3, 13);

ht.put(1, 15);

ht.put(2, 45);

System.out.println("Is Empty? : " + ht.isEmpty());

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

for (int i = 1; i <= ht.size(); i++) {

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

}

System.out.println();

ht.clear();

System.out.println("Is empty? : " + ht.isEmpty());

}

}

1. **Enumeration**

import java.util.\*;

public class EnumeratorDemo {

public static void main(String[] args) {

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

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

v.addElement(i);

}

Enumeration<Integer> e = v.elements();

while (e.hasMoreElements()) {

Integer i = (Integer) e.nextElement();

if (i % 2 == 0) {

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

}

}

System.out.println();

System.out.println(v);

}

}

1. **Iterator**

import java.util.Iterator;

import java.util.ArrayList;

public class IteratorDemo{

public static void main(String[] args){

ArrayList<Integer> l = new ArrayList<Integer>();

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

l.add(i);

Iterator itr = l.iterator();

while(itr.hasNext())

{

int x = (Integer)itr.next();

if(x==5)

{

System.out.println("5 is removed");

itr.remove();

}

else

System.out.println(x);

}

System.out.println(l);

}

}

1. **ListIterator**

import java.util.\*;

public class ListIteratorExample {

public static void main(String[] args) {

LinkedList<String> L = new LinkedList<>();

L.add("Ram"); L.add("Sita");

L.add("Vishnu");

L.add("Krishna"); L.add("Naruto");

System.out.println(L);

ListIterator<String> itr = L.listIterator();

while( itr.hasNext() ){

String s = (String)itr.next();

if( s.equals("Naruto") ){

itr.remove();

}

}

System.out.println(L);

// ListIterator<String> itr2 = L.listIterator();

while( itr.hasPrevious() ){

String s = (String)itr.previous();

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

if( s.equals("Sita") ){

itr.set("Bharat");

}

}

System.out.println();

System.out.println(L);

}

}

1. **Collection Sort**

import java.util.ArrayList;

import java.util.Collections;

public class SortMethod{

public static void main(String[] args){

ArrayList<Integer> l = new ArrayList<Integer>();

l.add(80);

l.add(90);

l.add(100);

l.add(50);

l.add(10);

l.add(70);

Collections.sort(l);

for(int x : l)

System.out.println(x);

}

}

1. **Comparable**

import java.util.ArrayList;

import java.util.Collections;

class Student implements Comparable<Student>{

int id;

String name;

int age;

Student(int id, String name, int age){

this.id = id;

this.name = name;

this.age = age;

}

public int compareTo(Student st){

if(age == st.age)

return 0;

else if(age > st.age)

return 1;

else

return -1;

}

public String toString(){

return id + " " + name + " " + age;

}

}

public class ComparableInterface{

public static void main(String[] args){

ArrayList<Student> l = new ArrayList<Student>();

l.add(new Student(50,"Shiva",46));

l.add(new Student(20,"Sreya",78));

l.add(new Student(10,"Neha",118));

l.add(new Student(40,"Apoorva",14));

l.add(new Student(30,"Santosh",33));

Collections.sort(l);

for(Student x : l)

System.out.println(x);

}

}

1. **Comparator**

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Collections;

class Student{

int id;

String name;

int age;

Student(int id, String name, int age){

this.id = id;

this.name = name;

this.age = age;

}

public String toString(){

return id + " " + name + " " + age;

}

}

class AgeComparator implements Comparator<Student>{

public int compare(Student s1,Student s2){

if(s1.age == s2.age)

return 0;

else if(s1.age > s2.age)

return 1;

else

return -1;

}

}

public class ComparatorInterface{

public static void main(String[] args){

ArrayList<Student> l = new ArrayList<Student>();

l.add(new Student(50,"Shiva",46));

l.add(new Student(20,"Sreya",78));

l.add(new Student(10,"Neha",118));

l.add(new Student(40,"Apoorva",14));

l.add(new Student(30,"Santosh",33));

Collections.sort(l,new AgeComparator());

for(Student x : l)

System.out.println(x);

}

}

1. **Write a Java program to create a file and write data into the file using Character Stream.**

import java.io.FileOutputStream;

import java.io.IOException;

public class FileCreation {

public static void main(String[] args) {

String fileContent = "Hello world";

FileOutputStream outputStream = null;

try {

outputStream = new FileOutputStream("file.txt");

byte[] strToBytes = fileContent.getBytes();

outputStream.write(strToBytes);

System.out.println("File is successfully created with the content");

}

catch(Exception e){

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

}

finally {

if(outputStream!=null){

try{

outputStream.close();

}

catch(IOException e){

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

}

}

}

}

}

1. **Write a Java program that reads on file name from the user then displays information about whether the file exists, whether the file is readable, whether the file is writable, the contents of file and the length of the file in bytes.**

import java.io.BufferedReader;

import java.io.File;

import java.io.IOException;

import java.io.InputStreamReader;

class FileDemo {

public static void p(String s){

System.out.println(s);

}

public static void analyze(String s){

File f = new File(s);

if(f.exists()){

p(f.getName() + " is a file");

p(f.canRead() ? "is readable":"is not readable");

p(f.canWrite()?"is writable":"is writable");

p("FileSize: " + f.length() + "bytes");

p("File last modified : " + f.lastModified());

}

if(f.isDirectory()){

p(f.getName() + " is a directory");

p("list of files");

String dir[] = f.list();

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

p(dir[i]);

}

}

}

}

public class FileDetails{

public static void main(String[] args) {

FileDemo fd = new FileDemo();

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

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

try{

String s = br.readLine();

fd.analyze(s);

}

catch (IOException e){

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

}

}

}

1. **Create a table in database using JDBC connection**

import java.sql.\*;

public class SqlTest {

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

System.out.println("\*\*\*Connection creation process\*\*\*");

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/teja","root","root");

Statement statement = con.createStatement();

System.out.println("\*\*\*Table creation process\*\*\*");

String query1 = "create table emp(eid varchar(50),ename varchar(50),sal int)";

int a = statement.executeUpdate(query1);

System.out.println("Table created successfully = " + a);

// System.out.println("\*\*\*Table dropped\*\*\*");

// String query2 = "drop table emp";

// int b = statement.executeUpdate(query2);

// System.out.println("table dropped succesfully");

}

}

1. **Insert data in a table using JDBC connection**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.ResultSet;

public class InsertDataDemo {

public static void main(String[] args) {

Connection con = null;

Statement statement = null;

try{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/teja","root","root");

System.out.println(con);

System.out.println("connection created successfully");

String query1 = "insert into emp values(111,'ratan',100000)";

String query2 = "insert into emp values(222,'anu',500000)";

String query3 = "insert into emp values(333,'shiva',1000000)";

statement = con.createStatement();

statement.executeUpdate(query1);

statement.executeUpdate(query2);

statement.executeUpdate(query3);

System.out.println("values inserted succesfully");

}

catch (ClassNotFoundException|SQLException e){

e.printStackTrace();

}

finally {

try{

if(statement!=null)

statement.close();

if(con!=null)

con.close();

}

catch (SQLException e){

e.printStackTrace();

}

}

System.out.println("connection closed successfully");

}

}

1. **CRUD operations on a table using JDBC connection**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class InsertValues2 {

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

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/csed","root","root");

Statement st = con.createStatement();

String query = "insert into first values(1, 'kamal')";

String query2 = "insert into first values(2, 'moses')";

String query3 = "insert into first values(3, 'arafah')";

int a = st.executeUpdate(query);

st.executeUpdate(query2);

st.executeUpdate(query3);

if(a > 0)

System.out.println("Values inserted successfully");

System.out.println("Updating the name as kamal where id =1");

query = "update first set name='mina' where id=1";

a = st.executeUpdate(query);

if(a > 0)

System.out.println("Updation done");

query = "select \* from first";

System.out.println("displaying the data");

ResultSet rs = st.executeQuery(query);

while(rs.next()){

System.out.println(rs.getInt("id") + "----" + rs.getString("name")) ;

}

System.out.println("deleting the details with name mina");

query = "delete from first where name='mina'";

a = st.executeUpdate(query);

if(a > 0)

System.out.println("deletion done successfully");

}

}