**SECTION 1**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to print ‘Welcome to Java’.** |

public class Welcome

{

public static void main(String aargs[])

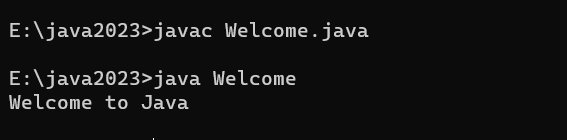
{

System.out.println("Welcome to Java");

}

}

**Output**

****

|  |
| --- |
| **Program 2 Date:** |
| **WAP to display two numbers received as command line argument, and print its product** |

import java.io.\*;

class Product

{

public static void main(String args[])

{

DataInputStream din;

int a,b;

String s;

din=new DataInputStream(System.in);

try

{

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

s=din.readLine();

a=Integer.parseInt(s);

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

s=din.readLine();

b=Integer.parseInt(s);

System.out.println("Product of " +a+ " and " +b+ " is: " +(a\*b));

}

catch(Exception e)

{

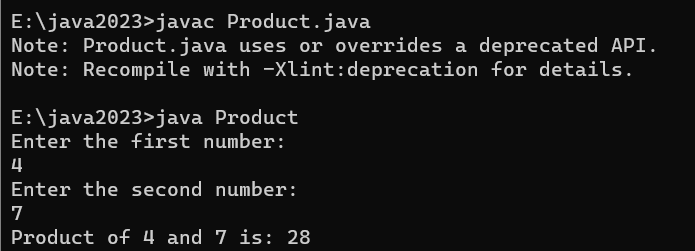
System.out.println("Error is:"+e);

}

}

}

**Output**

****

|  |
| --- |
| **Program 3 Date:** |
| **WAP to read two numbers and display the output in the form of ‘Sum of 2 and 3 is 5** |

import java.io.\*;

import java.lang.\*;

public class NumSum

{

public static void main(String aargs[])

{

try

{

String s;

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

DataInputStream dis=new DataInputStream(System.in);

s=dis.readLine();

int a=Integer.parseInt(s);

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

dis=new DataInputStream(System.in);

s=dis.readLine();

int b=Integer.parseInt(s);

int sum=a+b;

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

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

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

}

catch(Exception e)

{

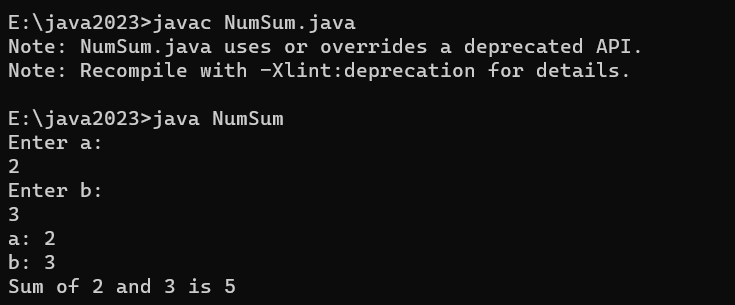
System.out.println(e);

}

}

}

**Output**

****

|  |
| --- |
| **Program 4 Date:** |
| **WAP to accept two numbers from the keyboard and swap them.** |

import java.io.\*;

public class SwapNum

{

public static void main(String args[])

{

try

{

String s;

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

DataInputStream dis=new DataInputStream(System.in);

s=dis.readLine();

int a=Integer.parseInt(s);

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

dis=new DataInputStream(System.in);

s=dis.readLine();

int b=Integer.parseInt(s);

System.out.println("Before swapping: ");

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

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

int t=a;

a=b;

b=t;

System.out.println("After swapping: ");

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

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

}

catch(Exception e)

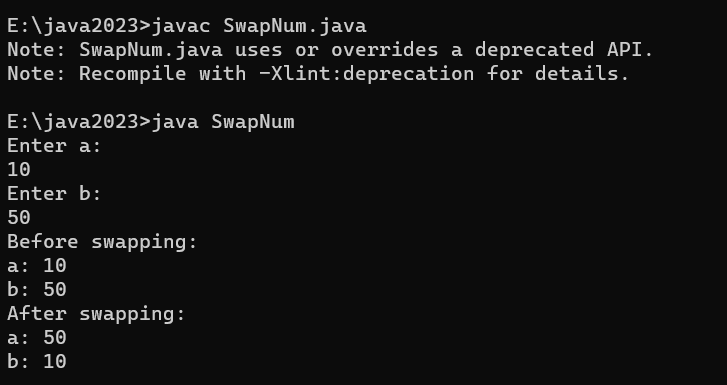
{

System.out.println(e);

}

}}

**Output**



**SECTION 2**

|  |
| --- |
| **Program 1 Date:** |
| **WAP to read three numbers and the maximum.** |

import java.io.\*;

import java.lang.\*;

public class MaxNum

{

public static void main(String args[])

{

try

{

String s;

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

DataInputStream dis=new DataInputStream(System.in);

s=dis.readLine();

int a=Integer.parseInt(s);

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

dis=new DataInputStream(System.in);

s=dis.readLine();

int b=Integer.parseInt(s);

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

dis=new DataInputStream(System.in);

s=dis.readLine();

int c=Integer.parseInt(s);

int max;

if(a>=b&&a>=c)

{

System.out.println("max: "+a);

}

else if(b>=a&&b>=c)

{

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

}

else

{

System.out.println("max: "+c);

}

}

catch(Exception e)

{

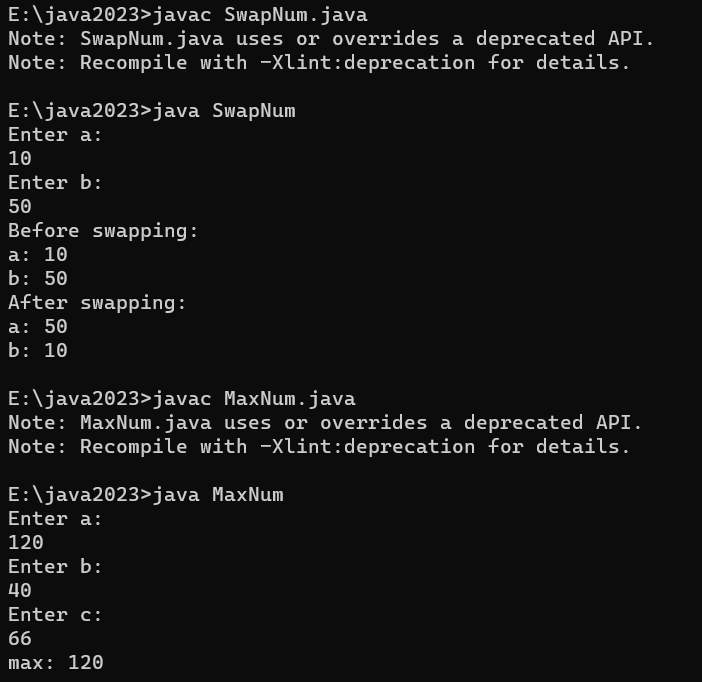
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Find the minimum of three numbers using a single statement.** |

import java.io.\*;

import java.lang.\*;

public class MinNum

{

public static void main(String args[])

{

try

{

String s;

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

DataInputStream dis=new DataInputStream(System.in);

s=dis.readLine();

int a=Integer.parseInt(s);

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

dis=new DataInputStream(System.in);

s=dis.readLine();

int b=Integer.parseInt(s);

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

dis=new DataInputStream(System.in);

s=dis.readLine();

int c=Integer.parseInt(s);

int min=Math.min(Math.min(a,b),c);

System.out.println("Min number is: "+min);

}

catch(Exception e)

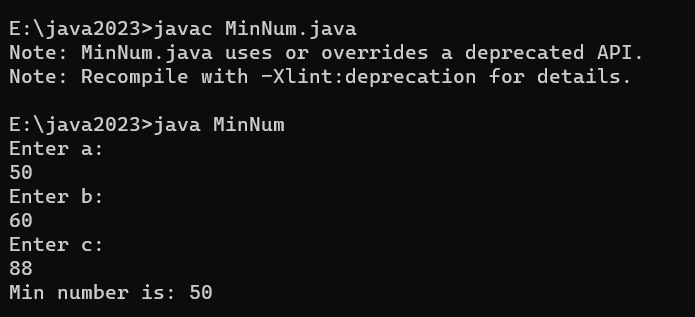
{

System.out.println(e);

}

}}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **WAP to search for a given element in an array.** |

import java.io.\*;

public class SearArr {

public static void main(String args[]) {

try {

int a[];

String s;

int i;

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

DataInputStream dis = new DataInputStream(System.in);

s = dis.readLine();

int n = Integer.parseInt(s);

a = new int[n];

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

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

s = dis.readLine();

a[i] = Integer.parseInt(s);

}

System.out.println("Array elements are: ");

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

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

}

System.out.println("Enter element to search: ");

s = dis.readLine();

int searchElement = Integer.parseInt(s);

int flag = 0;

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

if (a[i] == searchElement) {

flag = 1;

System.out.println("Element " + searchElement + " found at index " + i);

break;

}

}

if (flag == 0) {

System.out.println("Element " + searchElement + " not found in the array.");

}

} catch (Exception e) {

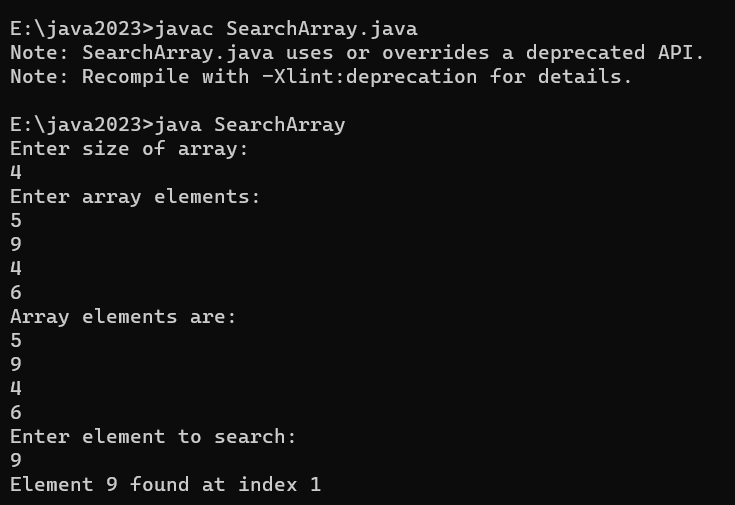
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **WAP to sort elements in an array in ascending order.** |

import java.io.\*;

public class SortArray {

public static void main(String args[]) {

try {

int a[];

String s;

int i;

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

DataInputStream dis = new DataInputStream(System.in);

s = dis.readLine();

int n = Integer.parseInt(s);

a = new int[n];

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

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

s = dis.readLine();

a[i] = Integer.parseInt(s);

}

System.out.println("Array elements before sorting: ");

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

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

}

bubbleSort(a);

System.out.println("Array elements after sorting in ascending order:");

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

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

}

} catch (Exception e) {

System.out.println(e);

}

}

static void bubbleSort(int[] arr) {

int n = arr.length;

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

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

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

int temp = arr[j];

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

arr[j + 1] = temp;

}

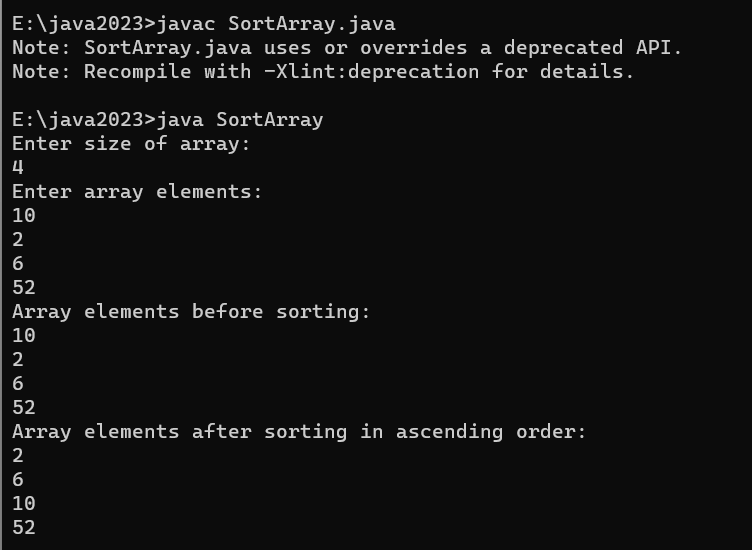
}

}

}

}

**Output**



|  |
| --- |
| **Program 5 Date:** |
| **Write a program to print the row wise and column wise sum of a 2D array.**  **1 2 3 | 6**  **2 1 1 |4**  **. . .**  **3 3 4** |

import java.io.\*;

public class ArraySum

{

public static void main(String[] args) {

DataInputStream i = new DataInputStream(System.in);

try {

int rows, cols;

int[][] array;

String s;

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

s = i.readLine();

rows = Integer.parseInt(s);

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

s = i.readLine();

cols = Integer.parseInt(s);

array = new int[rows][cols];

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

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

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

s = i.readLine();

array[j][k] = Integer.parseInt(s);

}

}

System.out.println("2D Array:");

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

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

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

}

System.out.println();

}

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

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

int rowSum = 0;

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

rowSum += array[j][k];

}

System.out.println("Row " + (j + 1) + ": " + rowSum);

}

System.out.println("Column-wise Sum:");

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

int colSum = 0;

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

colSum += array[j][k];

}

System.out.print("Column " + (k + 1) + ": " + colSum + " ");

}

System.out.println();

} catch (Exception e) {

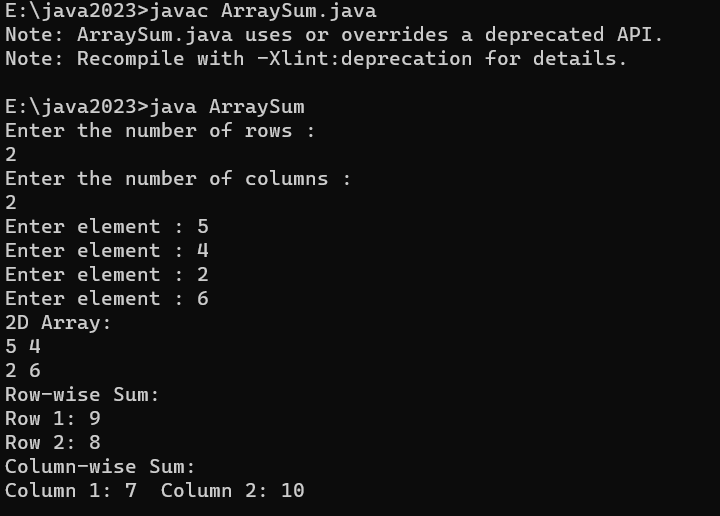
System.out.println("Error "+e);

}

}

}

**Output**



**SECTION 3**

|  |
| --- |
| **Program 1 Date:** |
| **WAP with two functions to check for an integer palindrome. (Function1 should reverse the integer. Function2 should return 1,if it is a palindrome or else 0.)** |

import java.io.\*;

public class PalindromeChecker {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

String input = dis.readLine();

int num = Integer.parseInt(input);

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

if (isPalindrome(num)) {

System.out.println("The number is a palindrome.");

} else {

System.out.println("The number is not a palindrome.");

}

} catch (Exception e) {

System.out.println(e);

}

}

public static int reverseInteger(int num) {

int reversed = 0;

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return reversed;

}

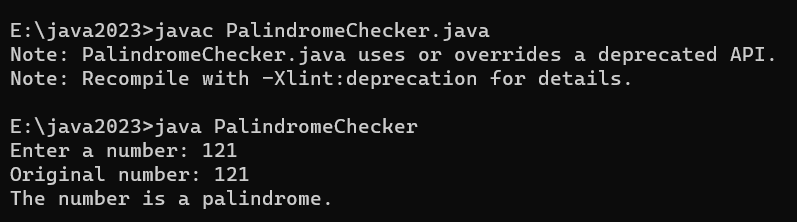
public static boolean isPalindrome(int num) {

return num == reverseInteger(num);

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **WAP to display numbers from m to n using single while loop.** |

import java.io.\*;

public class NumbersWhile {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

String mStr = dis.readLine();

int m = Integer.parseInt(mStr);

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

String nStr = dis.readLine();

int n = Integer.parseInt(nStr);

System.out.println("Numbers from " + m + " to " + n + ":");

while (m <= n) {

System.out.println(m);

m++;

}

} catch (Exception e) {

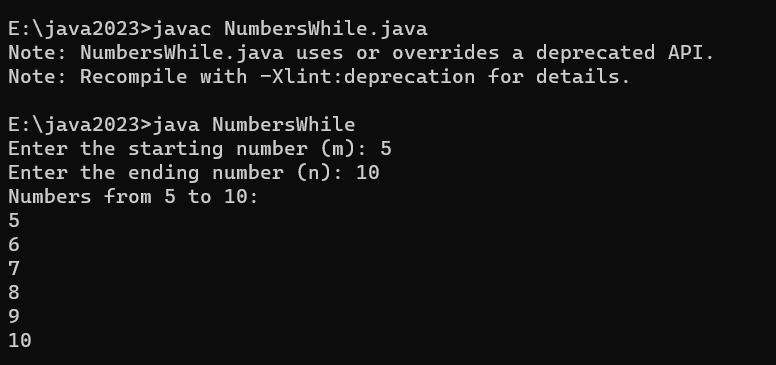
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **WAP to find the sum of the series 1+(1+2)+(1+2+3)+............ +(1+2+3+…+n) using a single while loop.** |

import java.io.\*;

public class SumSeries {

public static void main(String args[]) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int n = Integer.parseInt(dis.readLine());

int sum = 0;

int innerSum = 0;

int i = 1;

while (i <= n) {

innerSum += i;

sum += innerSum;

i++;

}

System.out.println("Sum of the series is: " + sum);

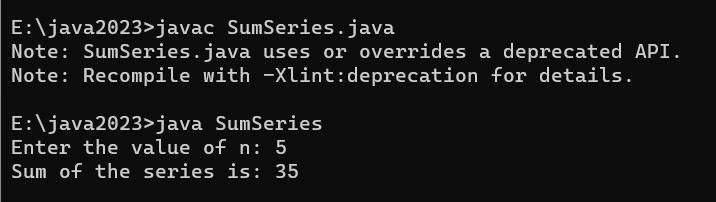
} catch (Exception e) {

System.out.println(e);

}

}}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **WAP to find the sum of 1+2/2!+3/3!+4/4!++n/n! using a single for loop.** |

import java.io.\*;

public class FactorialSum {

public static void main(String args[]) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int n = Integer.parseInt(dis.readLine());

double sum = 0;

double factorial = 1;

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

factorial \*= i;

sum += i / factorial;

}

System.out.println("Sum of the series is: " + sum);

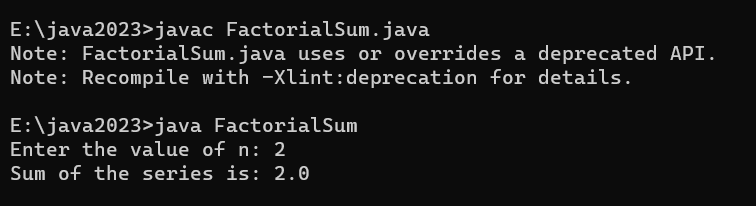
} catch (Exception e) {

System.out.println(e);

}

}}

**Output**



|  |
| --- |
| **Program 5 Date:** |
| **WAP to calculate area of a circle (functions with no argument and no return type.)** |

import java.io.\*;

public class CircleArea {

static void calculateArea() {

try {

DataInputStream dis = new DataInputStream(System.in);

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

double radius = Double.parseDouble(dis.readLine());

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

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

} catch (Exception e) {

System.out.println(e);

}

}

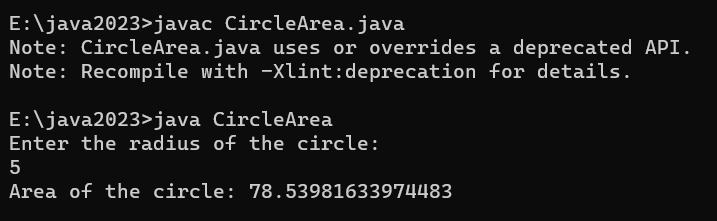
public static void main(String args[]) {

calculateArea();

}

}

**Output**



|  |
| --- |
| **Program 6 Date:** |
| **WAP to reverse a number (functions with argument and no return type.)** |

import java.io.\*;

public class ReverseNumber {

void reverseNumber(int number) {

int reversedNumber = 0;

while (number != 0) {

int digit = number % 10;

reversedNumber = reversedNumber \* 10 + digit;

number /= 10;

}

System.out.println("Reversed Number: " + reversedNumber);

}

public static void main(String args[]) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int inputNumber = Integer.parseInt(dis.readLine());

reverseNumber(inputNumber);

} catch (Exception e) {

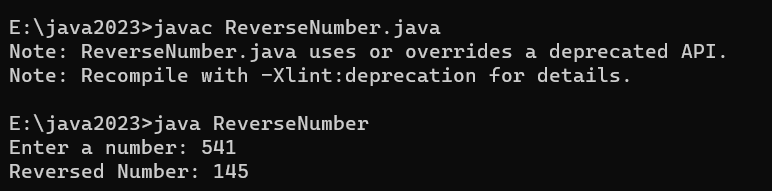
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 7 Date:** |
| **WAP to calculate sum of digits of a number (functions with argument and return type.)** |

import java.io.\*;

public class SumOfDigits {

static int calculateSumOfDigits(int number) {

int sum = 0;

while (number != 0) {

int digit = number % 10;

sum += digit;

number /= 10;

}

return sum;

}

public static void main(String args[]) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int inputNumber = Integer.parseInt(dis.readLine());

int sum = calculateSumOfDigits(inputNumber);

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

} catch (Exception e) {

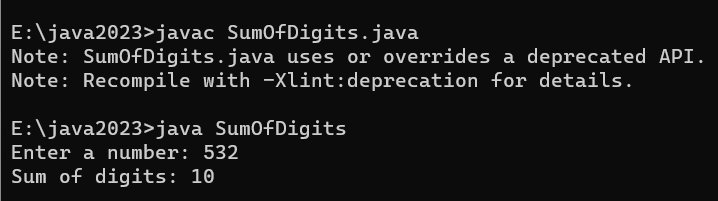
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 8 Date:** |
| **WAP to calculate sum of n even numbers (functions with no argument and return type.)** |

import java.io.\*;

public class SumOfEven {

static void calculateSumOfEvenNumbers() {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int n = Integer.parseInt(dis.readLine());

int sum = 0;

int count = 0;

int number = 2; // Starting from the first even number

while (count < n) {

sum += number;

number += 2; // Move to the next even number

count++;

}

System.out.println("Sum of the first " + n + " even numbers: " + sum);

} catch (Exception e) {

System.out.println(e);

}

}

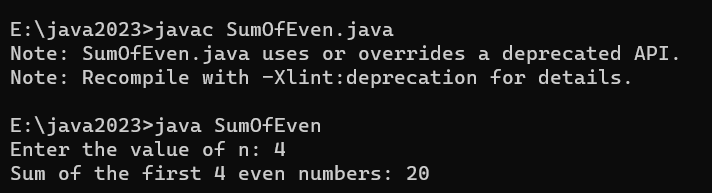
public static void main(String args[]) {

calculateSumOfEvenNumbers();

}

}

**Output**



**SECTION 4**

|  |
| --- |
| **Program 1 Date:** |
| **WAP with nested functions to find the maximum of three numbers. Function1 should take in two arguments and find the maximum. Function2 should take in the third number and the maximum from function1 to find the maximum.)** |

import java.io.\*;

public class MaxOfThreeNumbers {

static int findMax(int num1, int num2) {

return (num1 > num2) ? num1 : num2;

}

static int findMaxOfThree(int num1, int num2, int num3) {

int maxOfFirstTwo = findMax(num1, num2);

return findMax(maxOfFirstTwo, num3); }

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int num1 = Integer.parseInt(dis.readLine());

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

int num2 = Integer.parseInt(dis.readLine());

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

int num3 = Integer.parseInt(dis.readLine());

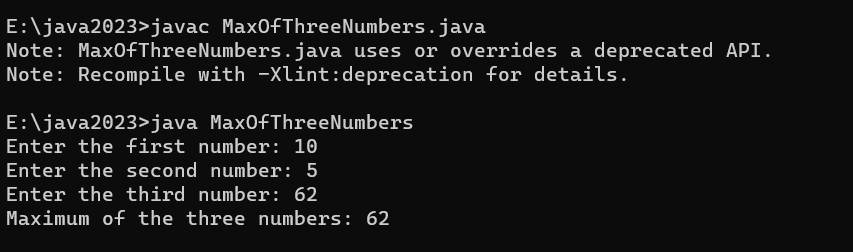
int max = findMaxOfThree(num1, num2, num3);

System.out.println("Maximum of the three numbers: " + max);

} catch (Exception e) {

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

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **WAP to find the factorial of n, using recursion.** |

import java.io.\*;

public class FactorialWithRecursion {

static int factorial(int n) {

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

return 1;

} else {

return n \* factorial(n - 1);

}

}

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int inputNumber = Integer.parseInt(dis.readLine());

int result = factorial(inputNumber);

System.out.println("Factorial of " + inputNumber + " is: " + result);

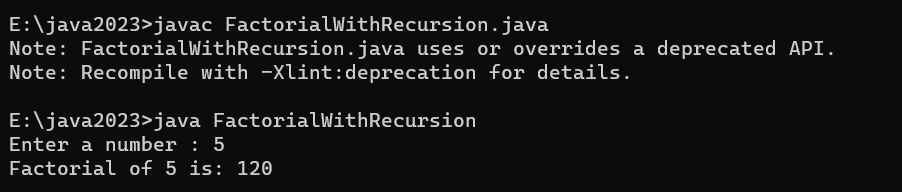
} catch (Exception e) {

System.out.println(e);

}

}}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **WAP to display numbers from n to 1 and vice versa, using recursion.** |

import java.io.\*;

public class DisplayNumbersRecursion {

static void displayDescending(int n) {

if (n >= 1) {

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

displayDescending(n - 1);

}

}

static void displayAscending(int n) {

if (n >= 1) {

displayAscending(n - 1);

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

}

}

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int inputNumber = Integer.parseInt(dis.readLine());

System.out.print("Descending Order: ");

displayDescending(inputNumber);

System.out.println(); // Move to the next line

System.out.print("Ascending Order: ");

displayAscending(inputNumber);

System.out.println(); // Move to the next line

} catch (Exception e) {

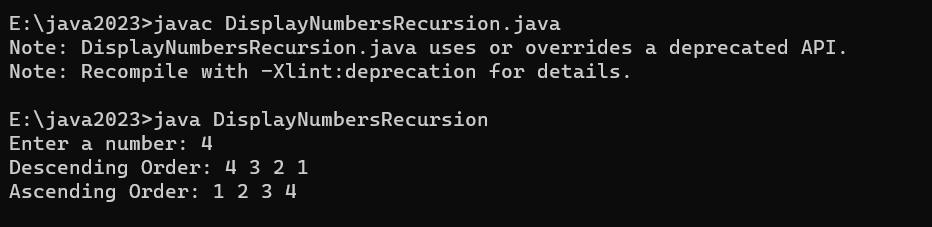
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **Using constructors, implement the operations of a queue.** |

import java.io.\*;

class Queue {

private static final int MAX\_SIZE = 5; // Maximum size of the queue

private int[] array;

private int front, rear, size;

public Queue() {

array = new int[MAX\_SIZE];

front = rear = -1;

size = 0;

}

public boolean isEmpty() {

return size == 0;

}

public boolean isFull() {

return size == MAX\_SIZE;

}

public void enqueue(int element) {

if (isFull()) {

System.out.println("Queue is full. Cannot enqueue.");

} else {

if (isEmpty()) {

front = 0;

}

rear = (rear + 1) % MAX\_SIZE;

array[rear] = element;

size++;

System.out.println("Enqueued: " + element);

}

}

public void dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty. Cannot dequeue.");

} else {

int removedElement = array[front];

if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % MAX\_SIZE;

}

size--;

System.out.println("Dequeued: " + removedElement);

}

}

public void display() {

if (isEmpty()) {

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

} else {

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

int i = front;

do {

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

i = (i + 1) % MAX\_SIZE;

} while (i != (rear + 1) % MAX\_SIZE);

System.out.println();

}

}

}

public class QueueImplementation {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

Queue myQueue = new Queue();

while (true) {

System.out.println("\nQueue Operations:");

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

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

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

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

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

int choice = Integer.parseInt(dis.readLine());

switch (choice) {

case 1:

System.out.print("Enter the element to enqueue: ");

int element = Integer.parseInt(dis.readLine());

myQueue.enqueue(element);

break;

case 2:

myQueue.dequeue();

break;

case 3:

myQueue.display();

break;

case 4:

System.out.println("Exiting the program.");

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

}

} catch (Exception e) {

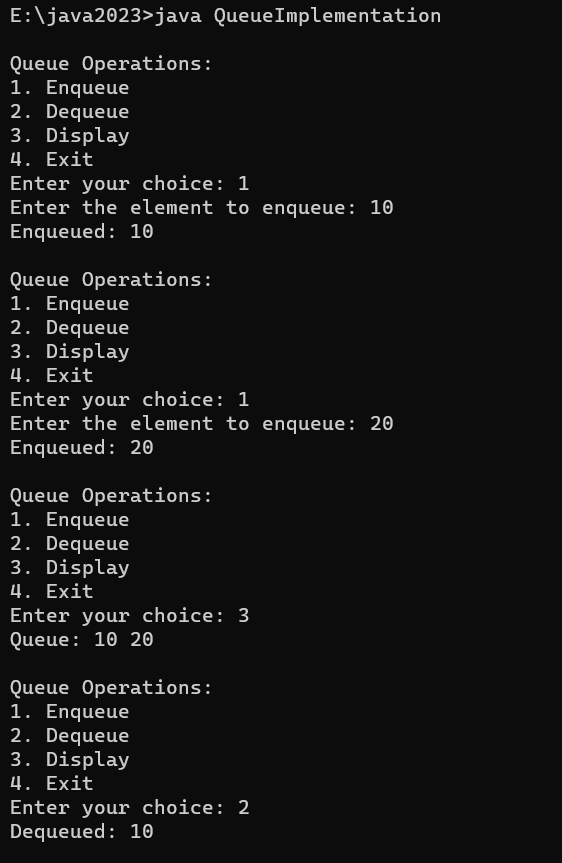
System.out.println(e);

}

}

}

**Output**

****

|  |
| --- |
| **Program 5 Date:** |
| **Create a class “weight” having a kg and gm part. Provide functions for read, display, add and subtracting two weights** |

import java.util.Scanner;

 class Weight {

    private int kg;

    private int gm;

    public Weight() {

        kg = 0;

        gm = 0;

    }

    public Weight(int kg, int gm) {

[this.kg](http://this.kg/) = kg;

[this.gm](http://this.gm/) = gm;

    }

    public void readWeight() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the weight in kg: ");

        kg = sc.nextInt();

        System.out.print("Enter the weight in gm: ");

        gm = sc.nextInt();

    }

    public void displayWeight() {

        System.out.println("Weight: " + kg + " kg " + gm + " gm");

    }

    public Weight addWeight(Weight w) {

        Weight sum = new Weight();

[sum.kg](http://sum.kg/) = kg + [w.kg](http://w.kg/);

[sum.gm](http://sum.gm/) = gm + [w.gm](http://w.gm/);

        if ([sum.gm](http://sum.gm/) >= 1000) {

[sum.kg](http://sum.kg/) += [sum.gm](http://sum.gm/) / 1000;

[sum.gm](http://sum.gm/) %= 1000;

        }

        return sum;

    }

    public Weight subtractWeight(Weight w) {

        Weight diff = new Weight();

        if (kg < [w.kg](http://w.kg/) || (kg == [w.kg](http://w.kg/) && gm < [w.gm](http://w.gm/))) {

            System.out.println("Error: Cannot subtract a larger weight from a smaller weight");

            return diff;

        }

[diff.kg](http://diff.kg/) = kg - [w.kg](http://w.kg/);

        if (gm < [w.gm](http://w.gm/)) {

            diff.kg--;

[diff.gm](http://diff.gm/) = gm + 1000 - [w.gm](http://w.gm/);

        } else {

[diff.gm](http://diff.gm/) = gm - [w.gm](http://w.gm/);

        }

        return diff;

    }

}

public class WeightOperations {

    public static void main(String[] args) {

        Weight w1 = new Weight();

        w1.readWeight();

        w1.displayWeight();

        Weight w2 = new Weight();

        w2.readWeight();

        w2.displayWeight();

        Weight sum = w1.addWeight(w2);

        System.out.print("Sum of weights: ");

        sum.displayWeight();

        Weight diff = w1.subtractWeight(w2);

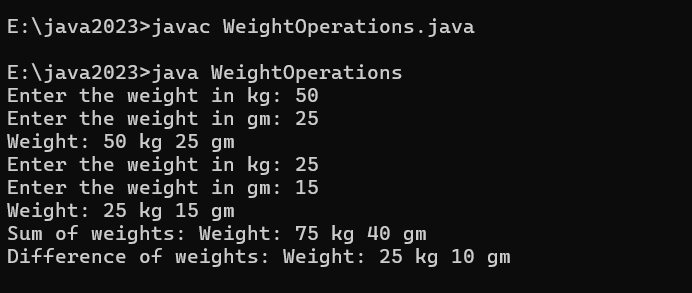
        System.out.print("Difference of weights: ");

        diff.displayWeight();

    }

}

**Output**



|  |
| --- |
| **Program 6 Date:** |
| **WAP to display even numbers upto ‘n’ using a static function.** |

import java.io.\*;

public class DisplayEvenNumber {

static void displayEvenNumbers(int n) {

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

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

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

}

System.out.println();

}

public static void main(String args[]) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int inputNumber = Integer.parseInt(dis.readLine());

displayEvenNumbers(inputNumber);

} catch (Exception e) {

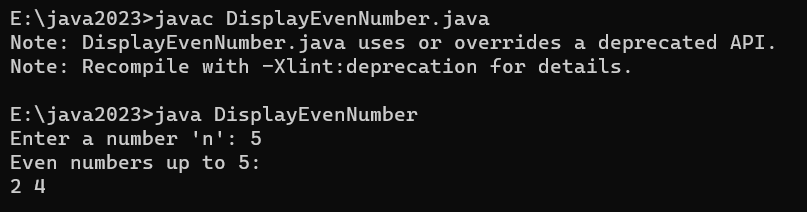
System.out.println(e);

}

}

}

**Output**



**SECTION 5**

|  |
| --- |
| **Program 1 Date:** |
| **WAP (menu driven) to demonstrate method overriding in java, by displaying details of a student, and a teacher.** |

import java.io.\*;

class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public void displayDetails() {

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

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

}

}

class Student extends Person {

private int rollNumber;

public Student(String name, int age, int rollNumber) {

super(name, age);

this.rollNumber = rollNumber;

}

public void displayDetails() {

super.displayDetails();

System.out.println("Roll Number: " + rollNumber);

}

}

class Teacher extends Person {

private String subject;

public Teacher(String name, int age, String subject) {

super(name, age);

this.subject = subject;

}

public void displayDetails() {

super.displayDetails();

System.out.println("Subject: " + subject);

}

}

public class MethodOverridingDemo {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Display Student Details");

System.out.println("2. Display Teacher Details");

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

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

int choice = Integer.parseInt(dis.readLine());

switch (choice) {

case 1:

Student student = new Student("James", 20, 43);

student.displayDetails();

break;

case 2:

Teacher teacher = new Teacher("Bindhu", 35, "Mathematics");

teacher.displayDetails();

break;

case 3:

System.out.println("Exiting the program.");

System.exit(0);

break;

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

}

} catch (Exception e) {

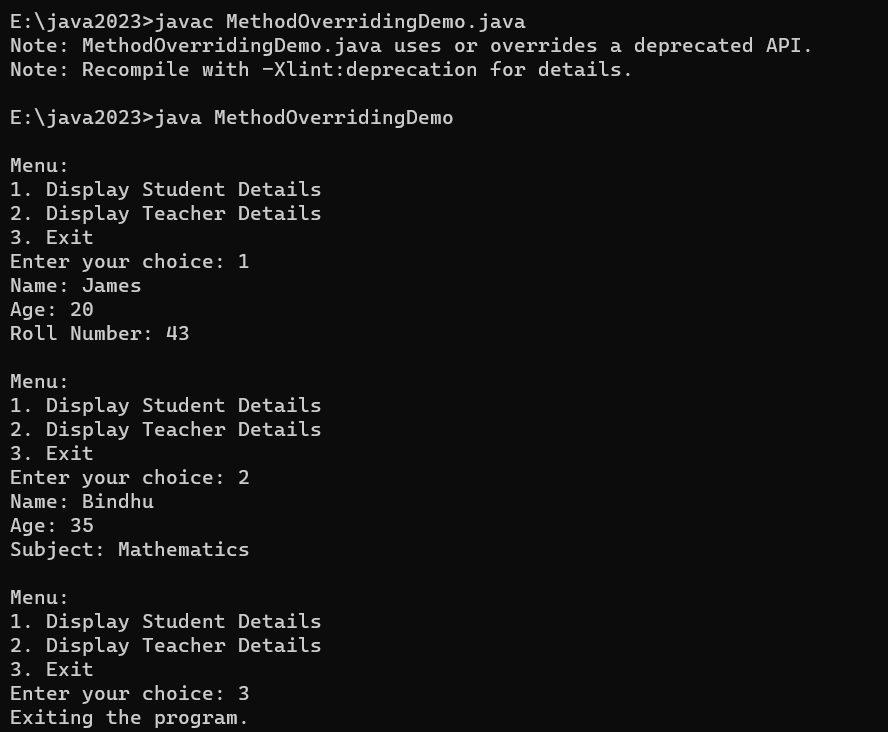
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Create a class for employee having eno,ename and esal as data members. Provide functions for reading and displaying employee details. (Accept information of n employees in the main function, display the same and search for an emp (using eno)).** |

import java.io.\*;

class Employee {

private int eno;

private String ename;

private double esal;

public void readDetails() {

try {

DataInputStream dis = new DataInputStream(System.in);

System.out.print("Enter Employee Number : ");

eno = Integer.parseInt(dis.readLine());

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

ename = dis.readLine();

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

esal = Double.parseDouble(dis.readLine());

} catch (Exception e) {

System.out.println(e);

}

}

public void displayDetails() {

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

System.out.println("Employee Number : " + eno);

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

System.out.println("Employee Salary : " + esal);

}

public boolean searchEmployee(int searchEno) {

return eno == searchEno;

}

}

public class EmployeeDetails {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int n = Integer.parseInt(dis.readLine());

Employee[] employees = new Employee[n];

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

employees[i] = new Employee();

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

employees[i].readDetails();

}

System.out.println("\nDetails of all Employees:");

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

employees[i].displayDetails();

System.out.println();

}

System.out.print("Enter Employee Number to search: ");

int searchEno = Integer.parseInt(dis.readLine());

boolean found = false;

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

if (employees[i].searchEmployee(searchEno)) {

System.out.println("Employee found!");

employees[i].displayDetails();

found = true;

break;

}

}

if (!found) {

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

}

} catch (Exception e) {

System.out.println(e);

}

}

}

**Output**



**SECTION 6**

|  |
| --- |
| **Program 1 Date:** |
| **Program to implement run time polymorphism in Java using interface, wrt calculating area of a triangle.** |

import java.io.\*;

interface Shape {

double calculateArea();

}

class Triangle implements Shape {

private double base;

private double height;

public Triangle(double base, double height) {

this.base = base;

this.height = height;

}

public double calculateArea() {

return 0.5 \* base \* height;

}

}

public class AreaTriangle {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

System.out.print("Enter the base of the triangle: ");

double base = Double.parseDouble(dis.readLine());

System.out.print("Enter the height of the triangle: ");

double height = Double.parseDouble(dis.readLine());

Triangle triangle = new Triangle(base, height);

System.out.println("Area of the triangle: " + triangle.calculateArea());

} catch (Exception e) {

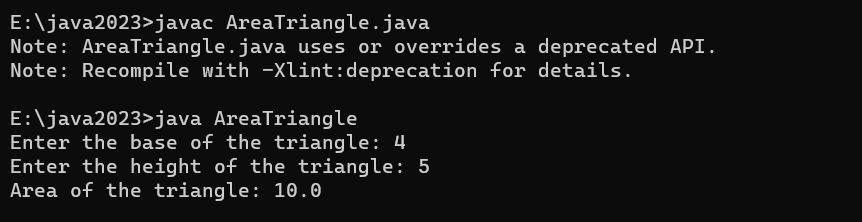
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Create an interface Shape having two prototypes disp() and calc(), to diplay the shape and calculate volume respectively. Create two classes circle and rectangle which implements the above interface. In the main function create a reference of Shape depending on the user-choice.** |

import java.io.\*;

interface Shape {

void disp();

void calc();

}

class Circle implements Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public void disp() {

System.out.println("Displaying Circle");

}

public void calc() {

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

System.out.println("Calculating Area of Circle: " + area);

}

}

class Rectangle implements Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public void disp() {

System.out.println("Displaying Rectangle");

}

public void calc() {

double area = length \* width;

System.out.println("Calculating Area of Rectangle: " + area);

}

}

public class InterfaceShape {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

System.out.println("Choose a shape:");

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

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

System.out.print("Enter your choice (1 or 2): ");

int choice = Integer.parseInt(dis.readLine());

Shape shape;

if (choice == 1) {

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

double radius = Double.parseDouble(dis.readLine());

shape = new Circle(radius);

} else if (choice == 2) {

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

double length = Double.parseDouble(dis.readLine());

System.out.print("Enter the width of the rectangle: ");

double width = Double.parseDouble(dis.readLine());

shape = new Rectangle(length, width);

} else {

System.out.println("Invalid choice. Exiting the program.");

return;

}

shape.disp();

shape.calc();

} catch (Exception e) {

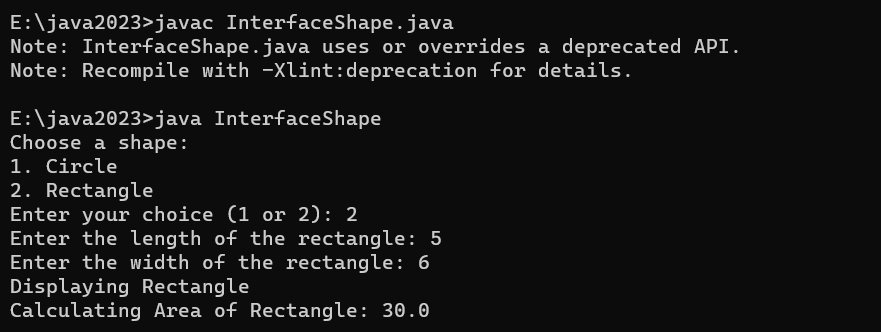
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **WAP to implement a function using call by value to swap two float numbers.** |

import java.io.\*;

public class SwapFloatNumbers

{

static void swap(float a, float b) {

System.out.println("Before swapping:");

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

System.out.println("b = " + b);

float temp = a;

a = b;

b = temp;

System.out.println("After swapping:");

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

System.out.println("b = " + b);

}

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

float a = Float.parseFloat(dis.readLine());

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

float b = Float.parseFloat(dis.readLine());

swap(a, b);

} catch (Exception e) {

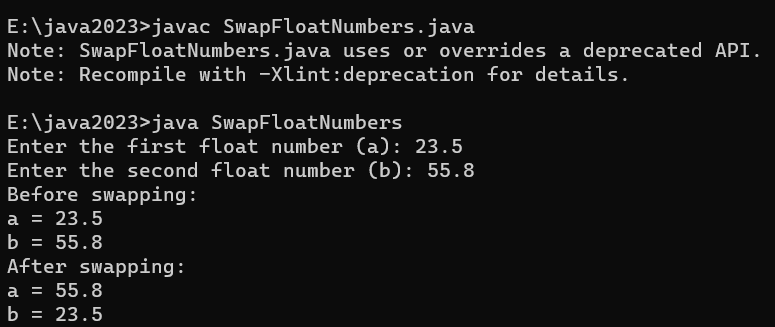
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **WAP to implement a function using call by reference to find the square root of a given number.** |

import java.io.\*;

public class SquareRoot {

static void findSquareRoot(double[] number) {

if (number[0] >= 0) {

number[0] = Math.sqrt(number[0]);

} else {

System.out.println("Cannot find square root of a negative number.");

}

}

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

double[] inputNumber = { Double.parseDouble(dis.readLine()) };

findSquareRoot(inputNumber);

System.out.println("Square root: " + inputNumber[0]);

} catch (Exception e) {

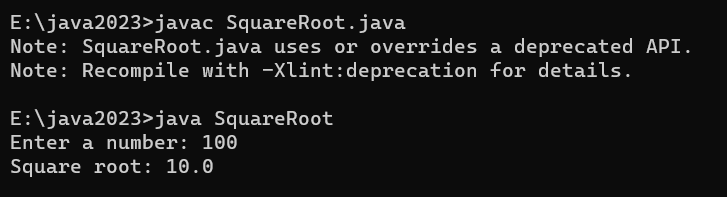
System.out.println(e);

}

}

}

**Output**



**SECTION 7**

|  |
| --- |
| **Program 1 Date:** |
| **Create a class for Cstring having a string data member and provide functions for read , display, compare(return Boolean value),add and concatenate.** |

import java.util.Scanner;

class Cstring {

static void display(String name1, String name2) {

System.out.println("Name1: " + name1);

System.out.println("Name2: " + name2);

}

static void compare(String name1, String name2) {

System.out.println(name1.equals(name2));

}

static void add(String name1, String name2) {

System.out.println(name1 + name2);

}

static void concatenate(String name1, String name2) {

System.out.println(name1.concat(name2));

}

public static void main(String args[]) {

try {

String name1, name2;

int c, ch = 1;

Scanner console = new Scanner(System.in);

System.out.println("First String: ");

name1 = console.nextLine();

System.out.println("Second String: ");

name2 = console.nextLine();

do {

System.out.println("1.Display\n2.Compare\n3.Add\n4.Concatenate");

c = console.nextInt();

switch (c) {

case 1:

display(name1, name2);

break;

case 2:

compare(name1, name2);

break;

case 3:

add(name1, name2);

break;

case 4:

concatenate(name1, name2);

break;

}

System.out.println("Do you want to continue? (1/0)");

ch = console.nextInt();

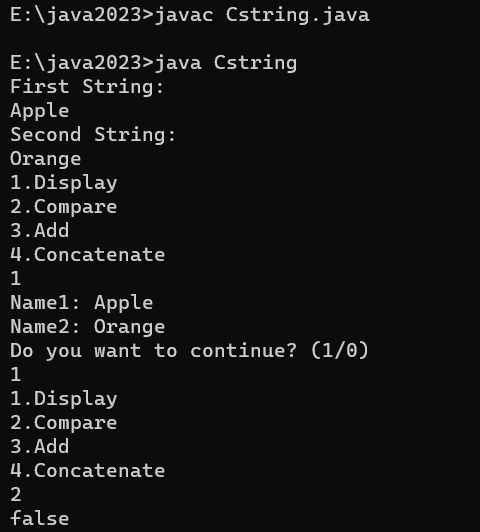
} while (ch == 1);

} catch (Exception e) {

e.printStackTrace(); }

}}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to implement object cloning for the class Distance which has inch and feet as data members.** |

import java.io.\*;

import java.lang.\*;

class Dist implements Cloneable{

Double inch,feet;

Dist(Double a,Double b){

inch=a;

feet=b;

}

public Object clone()

{

try

{

return super.clone();

}

catch(CloneNotSupportedException c){

System.out.println("error : "+c);

return this;}

}

}

class cloneDemo{

public static void main(String args[]){

Dist obj1= new Dist(9.1,43.0);

Dist obj2=(Dist)obj1.clone();

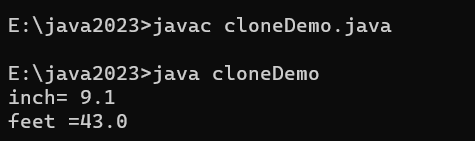
System.out.println("inch= "+obj2.inch);

System.out.println("feet ="+obj2.feet);

}

}

**Output**

****

|  |
| --- |
| **Program 3 Date:** |
| **Write a program to create a menu driven program for performing the following operations.**  **·        Length of a given string**  **·        Compare for equality**  **·        Extract a substring from a string.**  **·        Convert to uppercase and lowercase** |

import java.io.\*;

import java.util.Scanner;

import java.lang.\*;

class Cstr{

static void length(String name1,String name2){

System.out.println("Length of the first string is : "+name1.length());

System.out.println("Length of the second string is : "+name2.length());

}

static void compare(String name1,String name2){

System.out.println(name1==name2);

}

static void substr(String name1){

System.out.println("Substring of "+ name1+" is "+name1.substring(3));

}

static void lowercase(String name1,String name2){

System.out.println(name1.toLowerCase());

System.out.println(name2.toLowerCase());

}

public static void main(String args[]){

try{

String name1,name2;

int c,ch=1;

DataInputStream dir=new DataInputStream(System.in);

Scanner console=new Scanner(System.in);

System.out.println("First string : ");

name1 = console.nextLine();

System.out.println("Second string : ");

name2 = console.nextLine();

do{

System.out.println("1.Length\n2.Compare\n3.Substring\n4.Lowercase\n");

c=Integer.parseInt(dir.readLine());

switch(c){

case 1:

length(name1,name2);break;

case 2:

compare(name1,name2);break;

case 3:

substr(name1);break;

case 4:

lowercase(name1,name2);break;

}

System.out.println("Do you want to continue (1/0) ");

ch=Integer.parseInt(dir.readLine());

}while(ch==1);

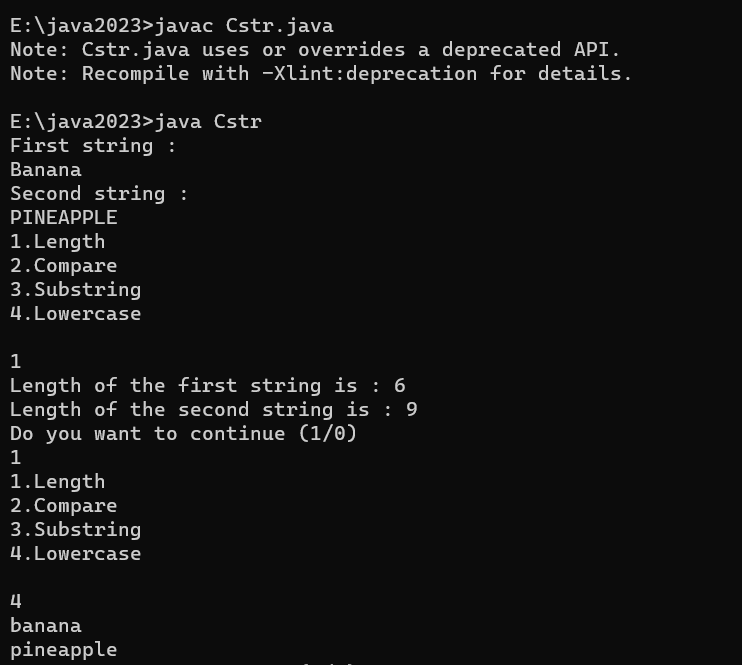
}

catch(Exception e){

System.out.println("error "+e);

}}}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **Write a program to reverse a string** |

import java.util.Scanner;

public class ReverseString {

public static String reverseString(String input) {

char[] charArray = input.toCharArray();

int start = 0;

int end = charArray.length - 1;

while (start < end) {

char temp = charArray[start];

charArray[start] = charArray[end];

charArray[end] = temp;

start++;

end--;

}

return new String(charArray);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

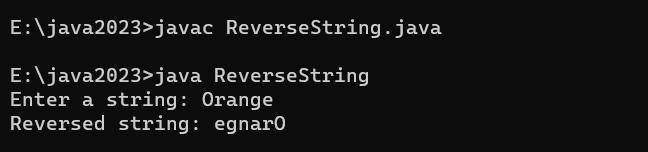
String inputString = scanner.nextLine();

String reversedString = reverseString(inputString);

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

}}

**Output**



|  |
| --- |
| **Program 5 Date:** |
| **Write a program to calculate the prime factors of a given number, using packages.** |

package package1;

import java.util.ArrayList;

import java.util.List;

public class PrimeFactorsCalculator {

public static List<Integer> calculatePrimeFactors(int number) {

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

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

while (number % i == 0) {

primeFactors.add(i);

number /= i;

}

}

return primeFactors;

}

}

import package1.PrimeFactorsCalculator;

import java.util.List;

import java.util.Scanner;

public class PrimeFactorsApp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number to find its prime factors: ");

int number = scanner.nextInt();

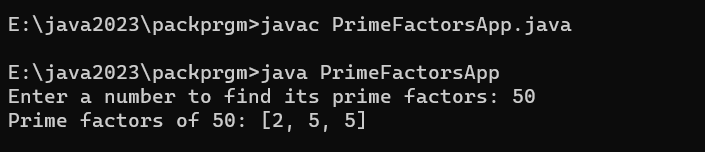
List<Integer> primeFactors = PrimeFactorsCalculator.calculatePrimeFactors(number);

System.out.println("Prime factors of " + number + ": " + primeFactors);

}

}

**Output**



|  |
| --- |
| **Program 6 Date:** |
| **Write a program to calculate the value of nCr for given value of n & r, using packages.** |

package package1;

public class CombinationCalculator {

private static long factorial(int n) {

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

return 1;

} else

{

return n \* factorial(n - 1);

}

}

public static long calculateCombination(int n, int r) {

if (n < r) {

return 0;

}

long numerator = factorial(n);

long denominator = factorial(r) \* factorial(n - r);

return numerator / denominator;

}

}

import package1.CombinationCalculator;

import java.util.Scanner;

public class CombinationApp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

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

int r = scanner.nextInt();

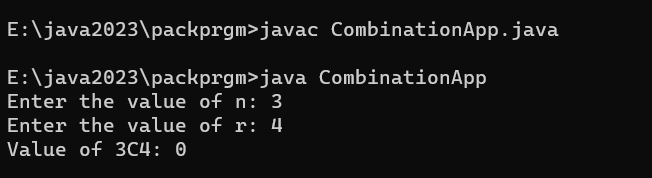
long result = CombinationCalculator.calculateCombination(n, r);

System.out.println("Value of " + n + "C" + r + ": " + result);

}

}

**Output**



**SECTION 8**

|  |
| --- |
| **Program 1 Date:** |
| **Read numbers into an array. Perform validations using multiple catch statements /** **predefined Exceptions.** |

import java.io.DataInputStream;

import java.io.IOException;

public class NumberArrayValidation {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(System.in);

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

int size = Integer.parseInt(dis.readLine());

if (size <= 0) {

throw new NegativeArraySizeException("Array size must be positive.");

}

int[] numbers = new int[size];

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

String[] inputNumbers = dis.readLine().split(" ");

if (inputNumbers.length != size) {

throw new IOException("Number of input numbers does not match the specified array size.");

}

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

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

}

System.out.println("The numbers you entered are:");

for (int number : numbers) {

System.out.println(number);

}

dis.close();

} catch (IOException e) {

System.out.println("IO Exception occurred.");

e.printStackTrace();

} catch (NumberFormatException e) {

System.out.println("Invalid input format. Please enter numbers separated by spaces.");

} catch (NegativeArraySizeException e) {

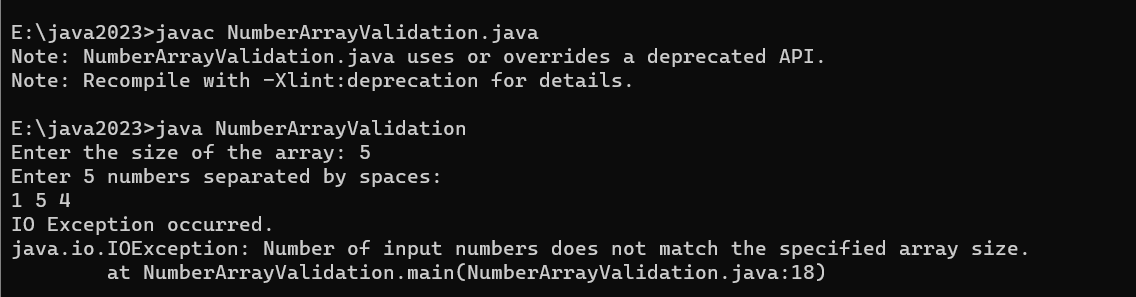
System.out.println("Negative array size entered. Please enter a positive size.");

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to implement a user defined Exception, which will throw an Exception when a given number is prime.** |

import java.util.Scanner;

class PrimeNumberException extends Exception {

public PrimeNumberException(String message) {

super(message);

}

}

public class PrimeChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter a number to check if it's prime: ");

int number = scanner.nextInt();

if (isPrime(number)) {

throw new PrimeNumberException(number + " is a prime number.");

} else {

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

}

} catch (PrimeNumberException e) {

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

} catch (Exception e) {

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

} finally {

scanner.close(); // Close the scanner to prevent resource leak

}

}

public static boolean isPrime(int number) {

if (number <= 1) {

return false;

}

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

if (number % i == 0) {

return false; // if the number is divisible by any other number, it's not prime

}

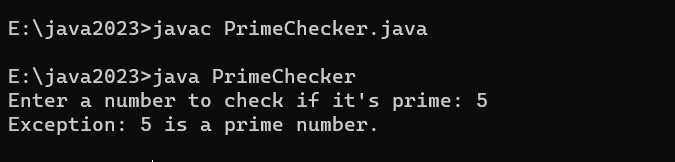
}

return true; // if the loop completes without finding a divisor, it's prime

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a program to implement throw and finally.** |

public class ThrowFinallyExample {

public static void main(String[] args) {

try {

int result = divide(10, 0);

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

} catch (ArithmeticException e) {

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

} finally {

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

}

}

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

if (num2 == 0) {

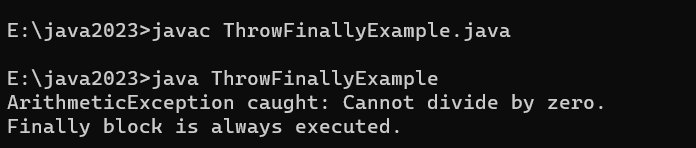
throw new ArithmeticException("Cannot divide by zero.");

}

return num1 / num2;

}}

**Output**



**SECTION 9**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to create multiple threads by extending the Thread class.** |

class MyThread extends Thread {

public void run() {

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

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

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

public class MultiThread {

public static void main(String[] args) {

MyThread thread1 = new MyThread();

MyThread thread2 = new MyThread();

MyThread thread3 = new MyThread();

thread1.setName("Thread 1");

thread2.setName("Thread 2");

thread3.setName("Thread 3");

thread1.start();

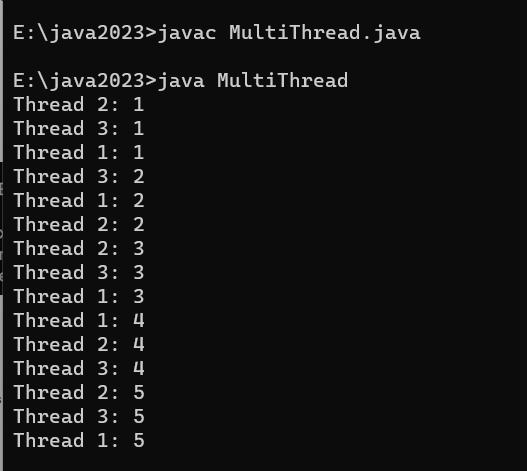
thread2.start();

thread3.start();

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to implement threads by implementing the Runnable interface.** |

class MyRunnable implements Runnable {

public void run() {

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

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

try {

Thread.sleep(1000); // Sleep for 1 second

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

public class threadRun {

public static void main(String[] args) {

MyRunnable myRunnable = new MyRunnable();

Thread thread1 = new Thread(myRunnable);

Thread thread2 = new Thread(myRunnable);

Thread thread3 = new Thread(myRunnable);

thread1.setName("Thread 1");

thread2.setName("Thread 2");

thread3.setName("Thread 3");

thread1.start();

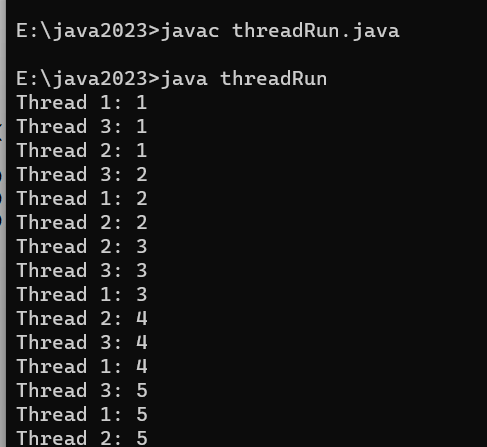
thread2.start();

thread3.start();

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a program to implement Synchronization using inter-thread communication** |

public class InterThreadCommunication {

public static void main(String[] args) {

SharedResource resource = new SharedResource();

Thread producerThread = new Thread(new Producer(resource));

Thread consumerThread = new Thread(new Consumer(resource));

producerThread.start();

consumerThread.start();

}

}

class SharedResource {

private int data;

private boolean available = false;

public synchronized void produce(int newData) {

while (available) {

try {

wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

data = newData;

available = true;

System.out.println("Produced: " + data);

notify();

}

public synchronized int consume() {

while (!available) {

try {

wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

available = false;

System.out.println("Consumed: " + data);

notify();

return data;

}

}

class Producer implements Runnable {

private final SharedResource resource;

public Producer(SharedResource resource) {

this.resource = resource;

}

public void run() {

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

resource.produce(i);

try {

Thread.sleep((long) (Math.random() \* 1000));

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

}

class Consumer implements Runnable {

private final SharedResource resource;

public Consumer(SharedResource resource) {

this.resource = resource;

}

public void run() {

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

resource.consume();

try {

Thread.sleep((long) (Math.random() \* 1000)); // simulate some work

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

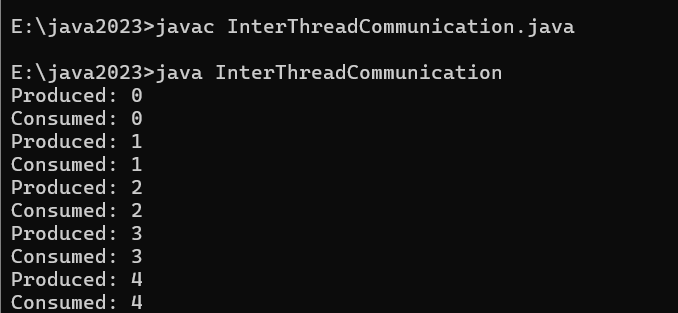
}

}

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **Implement the Producer- Consumer Problem, using Threads.** |

import java.lang.Thread.\*;

class Q {

int i;

int maxSize;

int producedCount = 0;

boolean produced = false;

Q(int maxSize) {

this.maxSize = maxSize; }

synchronized void produce(int x) {

if (produced) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

} }

i = x;

produced = true;

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

producedCount++;

notify();

}

synchronized int consume() {

if (!produced) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

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

produced = false;

notify();

return i;

}

}

class Producer extends Thread {

Q q;

int produceLimit;

Producer(Q q, int produceLimit) {

this.q = q;

this.produceLimit = produceLimit;

}

public void run() {

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

q.produce(i);

}

}

}

class Consumer extends Thread {

Q q;

int consumeLimit;

Consumer(Q q, int consumeLimit) {

this.q = q;

this.consumeLimit = consumeLimit;

}

public void run() {

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

q.consume();

}

}

}

public class ProducerConsumer {

public static void main(String args[]) {

int maxSize = 10;

int produceLimit = 10;

int consumeLimit = 10;

Q q = new Q(maxSize);

Producer p = new Producer(q, produceLimit);

Consumer c = new Consumer(q, consumeLimit);

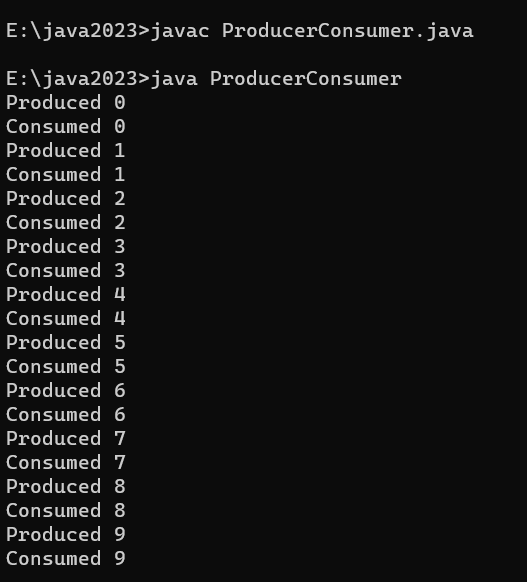
p.start();

c.start();

}

}

**Output**



**SECTION 10**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to display the contents of a directory by displaying the subdirectory’s name first, then the file names.** |

import java.io.File;

public class DirectoryContents {

public static void main(String[] args) {

String directoryPath = "E:/java2023";

File directory = new File(directoryPath);

if (directory.exists() && directory.isDirectory()) {

displayContents(directory);

} else {

System.out.println("Invalid directory path.");

}

}

public static void displayContents(File directory) {

File[] files = directory.listFiles();

for (File file : files) {

if (file.isDirectory()) {

System.out.println("Subdirectory: " + file.getName());

}

}

for (File file : files) {

if (file.isFile()) {

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

}

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to display the contents of a directory including its subdirectory content using recursive function.** |

import java.io.File;

public class DirectoryContentsRecursive {

public static void main(String[] args) {

String directoryPath = "E:/java2023";

File directory = new File(directoryPath);

if (directory.exists() && directory.isDirectory()) {

displayContentsRecursive(directory, 0);

} else {

System.out.println("Invalid directory path.");

}

}

public static void displayContentsRecursive(File directory, int depth) {

File[] files = directory.listFiles();

if (files != null) {

for (File file : files) {

if (file.isDirectory()) {

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

System.out.print("\t");

}

System.out.println("Subdirectory: " + file.getName());

displayContentsRecursive(file, depth + 1);

}

}

for (File file : files) {

if (file.isFile()) {

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

System.out.print("\t");

}

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

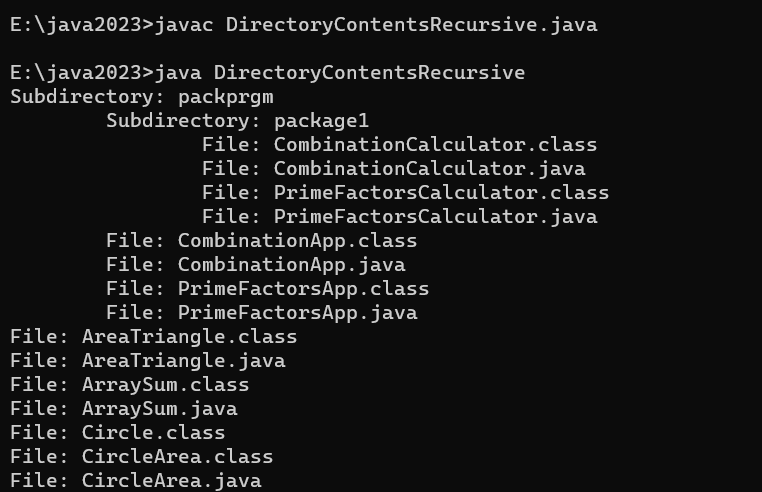
}

}

}

}}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a program to search for a given file name in a directory (including its subdirectory) content.** |

**import java.io.File;**

**public class FileSearch {**

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

**String rootDirectoryPath = "E://java2023";**

**String targetFileName = "Welcome.java";**

**File rootDirectory = new File(rootDirectoryPath);**

**if (rootDirectory.exists() && rootDirectory.isDirectory()) {**

**searchFile(rootDirectory, targetFileName);**

**} else {**

**System.out.println("Invalid directory path or directory does not exist.");**

**}**

**}**

**private static void searchFile(File directory, String targetFileName) {**

**File[] files = directory.listFiles();**

**if (files != null) {**

**for (File file : files) {**

**if (file.isDirectory()) {**

**searchFile(file, targetFileName); // Recursively search in subdirectories**

**} else if (file.getName().equals(targetFileName)) {**

**System.out.println("Found file: " + file.getAbsolutePath());**

**}**

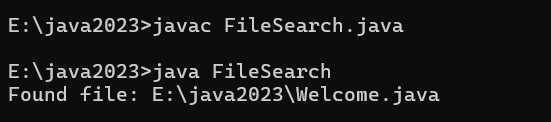
**}**

**}**

**}**

**}**

**Output**

****

**SECTION 11**

|  |
| --- |
| **Program 1 Date:** |
| **Write a menu driven program to demonstrate Random Access File handling, with options for creating, deleting, writing, appending and reading the file.** |

import java.io.File;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.util.Scanner;

public class RandomAccessFileDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Create a file");

System.out.println("2. Delete the file");

System.out.println("3. Write to the file");

System.out.println("4. Append to the file");

System.out.println("5. Read from the file");

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

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

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

createFile(scanner);

break;

case 2:

deleteFile(scanner);

break;

case 3:

writeToFile(scanner);

break;

case 4:

appendToFile(scanner);

break;

case 5:

readFromFile(scanner);

break;

case 6:

scanner.close();

System.out.println("Exiting program.");

System.exit(0);

default:

System.out.println("Invalid choice. Please enter a number between 1 and 6.");

}

}

}

private static void createFile(Scanner scanner) {

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

String fileName = scanner.nextLine();

try {

File file = new File(fileName);

if (file.createNewFile()) {

System.out.println("File created: " + file.getAbsolutePath());

} else {

System.out.println("File already exists.");

}

} catch (IOException e) {

System.out.println("An error occurred while creating the file: " + e.getMessage());

}

}

private static void deleteFile(Scanner scanner) {

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

String fileName = scanner.nextLine();

File file = new File(fileName);

if (file.delete()) {

System.out.println("File deleted: " + file.getAbsolutePath());

} else {

System.out.println("Failed to delete the file or the file does not exist.");

}

}

private static void writeToFile(Scanner scanner) {

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

String fileName = scanner.nextLine();

RandomAccessFile raf = null;

try {

raf = new RandomAccessFile(fileName, "rw");

System.out.print("Enter the content to write: ");

String content = scanner.nextLine();

raf.writeBytes(content);

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

} catch (IOException e) {

System.out.println("An error occurred while writing to the file: " + e.getMessage());

} finally {

if (raf != null) {

try {

raf.close();

} catch (IOException e) {

System.out.println("An error occurred while closing the file: " + e.getMessage());

}

}

}

}

private static void appendToFile(Scanner scanner) {

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

String fileName = scanner.nextLine();

RandomAccessFile raf = null;

try {

raf = new RandomAccessFile(fileName, "rw");

System.out.print("Enter the content to append: ");

String content = scanner.nextLine();

raf.seek(raf.length());

raf.writeBytes(content);

System.out.println("Content appended to the file.");

} catch (IOException e) {

System.out.println("An error occurred while appending to the file: " + e.getMessage());

} finally {

if (raf != null) {

try {

raf.close();

} catch (IOException e) {

System.out.println("An error occurred while closing the file: " + e.getMessage());

}

}

}

}

private static void readFromFile(Scanner scanner) {

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

String fileName = scanner.nextLine();

RandomAccessFile raf = null;

try {

raf = new RandomAccessFile(fileName, "r");

System.out.println("Content of the file:");

String line;

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

System.out.println(line);

}

} catch (IOException e) {

System.out.println("An error occurred while reading from the file: " + e.getMessage());

} finally {

if (raf != null) {

try {

raf.close();

} catch (IOException e) {

System.out.println("An error occurred while closing the file: " + e.getMessage());

}

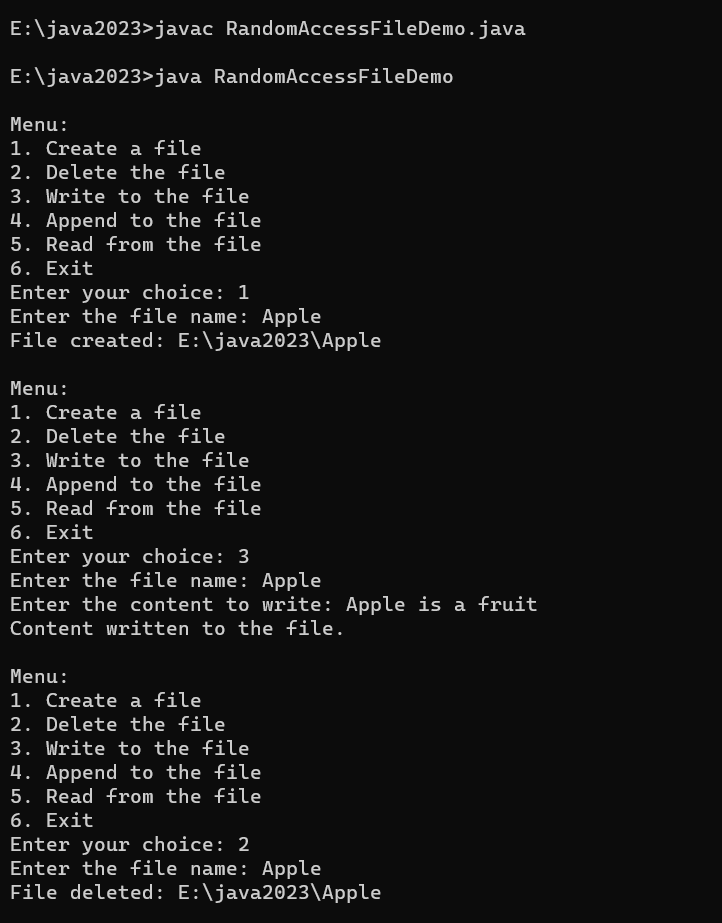
}

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to implement a Generic method, which can display the elements of various arrays of different data types, and find the length of each array.** |

public class GenericArrayDemo

{

public static <T> void displayAndFindLength(T[] array)

{

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

for (T element : array) {

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

}

System.out.println("\nLength of the array: " + array.length);

}

public static void main(String[] args)

{

Integer[] intArray = {1, 2, 3, 4, 5};

Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5};

Character[] charArray = {'a', 'b', 'c', 'd', 'e'};

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

System.out.println("Integer Array:");

displayAndFindLength(intArray);

System.out.println("\nDouble Array:");

displayAndFindLength(doubleArray);

System.out.println("\nCharacter Array:");

displayAndFindLength(charArray);

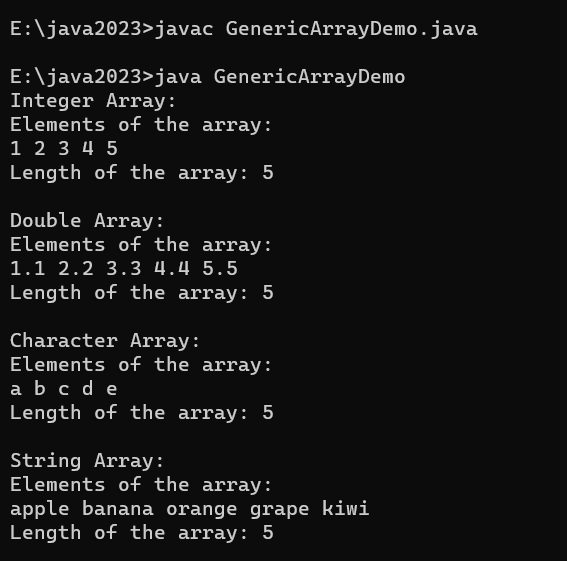
System.out.println("\nString Array:");

displayAndFindLength(stringArray);

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a program to implement a Generic class, and display the types of various parameters passed.** |

public class GenericClassDemo<T> {

public void displayType(T parameter) {

System.out.println("Type of the parameter: " + parameter.getClass().getName());

}

public static void main(String[] args) {

GenericClassDemo<Integer> integerInstance = new GenericClassDemo<Integer>();

GenericClassDemo<Double> doubleInstance = new GenericClassDemo<Double>();

GenericClassDemo<String> stringInstance = new GenericClassDemo<String>();

GenericClassDemo<Boolean> booleanInstance = new GenericClassDemo<Boolean>();

integerInstance.displayType(10);

doubleInstance.displayType(3.14);

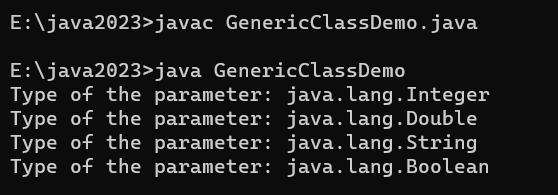
stringInstance.displayType("Hello");

booleanInstance.displayType(true);

}

}

**Output**

****

|  |
| --- |
| **Program 4 Date:** |
| **Using bounded type parameters, compare the averages of various arrays.** |

public class ArrayAverageCompare {

public static <T extends Number> double computeAverage(T[] array) {

double sum = 0;

for (T element : array) {

sum += element.doubleValue();

}

return sum / array.length;

}

public static <T extends Number> void compareAverages(T[] array1, T[] array2) {

double average1 = computeAverage(array1);

double average2 = computeAverage(array2);

System.out.println("Average of array1: " + average1);

System.out.println("Average of array2: " + average2);

if (average1 > average2) {

System.out.println("Average of array1 is greater than average of array2.");

} else if (average1 < average2) {

System.out.println("Average of array1 is less than average of array2.");

} else {

System.out.println("Average of array1 is equal to average of array2.");

}

}

public static void main(String[] args) {

Integer[] intArray1 = {10, 20, 30, 40, 50};

Integer[] intArray2 = {15, 25, 35, 45, 55};

Double[] doubleArray1 = {1.5, 2.5, 3.5, 4.5, 5.5};

Double[] doubleArray2 = {2.0, 3.0, 4.0, 5.0, 6.0};

System.out.println("Comparing averages of integer arrays:");

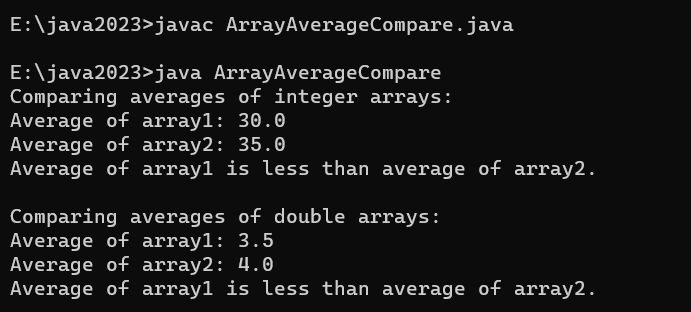
compareAverages(intArray1, intArray2);

System.out.println("\nComparing averages of double arrays:");

compareAverages(doubleArray1, doubleArray2);

}}

**Output**



**SECTION 12**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to implement Serialization and De-Serialization, for an object of Student Class.** |

import java.io.\*;

import java.util.Scanner;

class Student implements Serializable {

private static final long serialVersionUID = 1L;

private String name;

private int id;

public Student(String name, int id) {

this.name = name;

this.id = id;

}

public String getName() {

return name;

}

public int getId() {

return id;

}

public String toString() {

return "Student [name=" + name + ", id=" + id + "]";

}}

public class SerializationDemoUserInput {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Getting student details from user

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

String name = scanner.nextLine();

System.out.println("Enter student ID:");

int id = scanner.nextInt();

Student student = new Student(name, id);

// Serialization

try {

FileOutputStream fileOut = new FileOutputStream("student.ser");

ObjectOutputStream out = new ObjectOutputStream(fileOut);

out.writeObject(student);

out.close();

fileOut.close();

System.out.println("Serialized data is saved in student.ser");

} catch (IOException i) {

i.printStackTrace();

}

Student deserializedStudent = null;

try {

FileInputStream fileIn = new FileInputStream("student.ser");

ObjectInputStream in = new ObjectInputStream(fileIn);

deserializedStudent = (Student) in.readObject();

in.close();

fileIn.close();

} catch (IOException i) {

i.printStackTrace();

return;

} catch (ClassNotFoundException c) {

System.out.println("Student class not found");

c.printStackTrace();

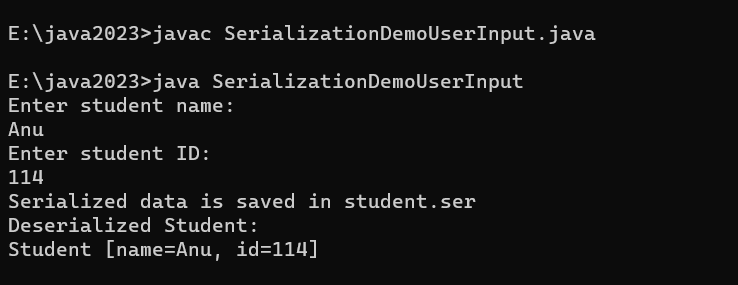
return;

} System.out.println("Deserialized Student:");

System.out.println(deserializedStudent);

scanner.close(); }}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to implement IS-A Serialization and De-Serialization, for a Maruti Car inherited from Vehicle.** |

import java.io.\*;

import java.util.Scanner;

// Base class (parent)

class Vehicle implements Serializable {

private static final long serialVersionUID = 1L;

private String make;

private String model;

public Vehicle(String make, String model) {

this.make = make;

this.model = model;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

@Override

public String toString() {

return "Vehicle [make=" + make + ", model=" + model + "]";

}

}

// Derived class (child)

class MarutiCar extends Vehicle implements Serializable {

private static final long serialVersionUID = 1L;

private int year;

public MarutiCar(String make, String model, int year) {

super(make, model);

this.year = year;

}

public int getYear() {

return year;

}

public String toString() {

return "MarutiCar [make=" + getMake() + ", model=" + getModel() + ", year=" + year + "]";

}

}

public class SerializationDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Getting Maruti car details from user

System.out.println("Enter Maruti car make:");

String make = scanner.nextLine();

System.out.println("Enter Maruti car model:");

String model = scanner.nextLine();

System.out.println("Enter Maruti car year:");

int year = scanner.nextInt();

MarutiCar car = new MarutiCar(make, model, year);

// Serialization

try {

FileOutputStream fileOut = new FileOutputStream("maruti\_car.ser");

ObjectOutputStream out = new ObjectOutputStream(fileOut);

out.writeObject(car);

out.close();

fileOut.close();

System.out.println("Serialized data is saved in maruti\_car.ser");

} catch (IOException i) {

i.printStackTrace();

}

// Deserialization

MarutiCar deserializedCar = null;

try {

FileInputStream fileIn = new FileInputStream("maruti\_car.ser");

ObjectInputStream in = new ObjectInputStream(fileIn);

deserializedCar = (MarutiCar) in.readObject();

in.close();

fileIn.close();

} catch (IOException i) {

i.printStackTrace();

return;

} catch (ClassNotFoundException c) {

System.out.println("MarutiCar class not found");

c.printStackTrace();

return;

}

// Displaying deserialized car

System.out.println("Deserialized MarutiCar:");

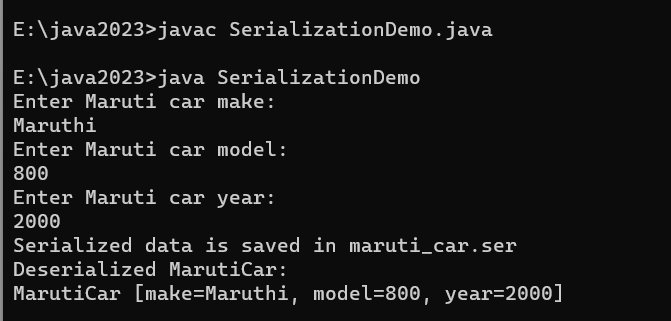
System.out.println(deserializedCar);

scanner.close();

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a program to implement HAS-A Serialization and De-Serialization for the Engine of a Vehicle.** |

import java.io.\*;

import java.util.Scanner;

class Engine implements Serializable {

private static final long serialVersionUID = 1L;

private String type;

private double horsepower;

public Engine(String type, double horsepower) {

this.type = type;

this.horsepower = horsepower;

}

public String getType() {

return type;

}

public double getHorsepower() {

return horsepower;

}

public String toString() {

return "Engine [type=" + type + ", horsepower=" + horsepower + "]";

}

}

class Vehicle implements Serializable {

private static final long serialVersionUID = 1L;

private String make;

private String model;

private transient Engine engine;

public Vehicle(String make, String model, Engine engine) {

this.make = make;

this.model = model;

this.engine = engine;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public Engine getEngine() {

return engine;

}

public String toString() {

return "Vehicle [make=" + make + ", model=" + model + ", engine=" + engine + "]";

}

}

public class SerializationDemoUserInput2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter engine type:");

String type = scanner.nextLine();

System.out.println("Enter engine horsepower:");

double horsepower = scanner.nextDouble();

scanner.nextLine();

Engine engine = new Engine(type, horsepower);

System.out.println("Enter vehicle make:");

String make = scanner.nextLine();

System.out.println("Enter vehicle model:");

String model = scanner.nextLine();

Vehicle vehicle = new Vehicle(make, model, engine);

try {

FileOutputStream fileOut = new FileOutputStream("vehicle.ser");

ObjectOutputStream out = new ObjectOutputStream(fileOut);

out.writeObject(vehicle);

out.close();

fileOut.close();

System.out.println("Serialized data is saved in vehicle.ser");

} catch (IOException i) {

i.printStackTrace();

}

Vehicle deserializedVehicle = null;

try {

FileInputStream fileIn = new FileInputStream("vehicle.ser");

ObjectInputStream in = new ObjectInputStream(fileIn);

deserializedVehicle = (Vehicle) in.readObject();

in.close();

fileIn.close();

} catch (IOException i) {

i.printStackTrace();

return;

} catch (ClassNotFoundException c) {

System.out.println("Vehicle class not found");

c.printStackTrace();

return;

}

System.out.println("Deserialized Vehicle:");

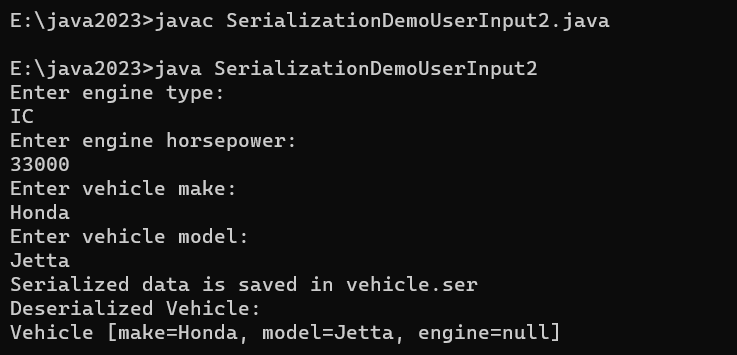
System.out.println(deserializedVehicle);

scanner.close();

}

}

**Output**

****

|  |
| --- |
| **Program 4 Date:** |
| **Write a program to Serialize/De-Serialize selected attributes of an Employee.** |

import java.io.\*;

import java.util.Scanner;

class Employee implements Serializable {

private static final long serialVersionUID = 1L;

private String name;

private int age;

private double salary;

private transient String address; // This attribute won't be serialized

public Employee(String name, int age, double salary, String address) {

this.name = name;

this.age = age;

this.salary = salary;

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String name = scanner.nextLine();

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

int age = scanner.nextInt();

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

double salary = scanner.nextDouble();

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

// Consume newline character after reading the double to prevent skipping next line

scanner.nextLine();

String address = scanner.nextLine();

// Create an Employee object

Employee employee = new Employee(name, age, salary, address);

// Serialize selected attributes

try {

FileOutputStream fileOut = new FileOutputStream("employee.ser");

ObjectOutputStream out = new ObjectOutputStream(fileOut);

// Write only selected attributes

out.writeObject(employee.getName());

out.writeObject(employee.getAge());

out.writeObject(employee.getSalary());

out.close();

fileOut.close();

System.out.println("Serialized data is saved in employee.ser");

} catch (IOException ioe) {

ioe.printStackTrace();

}

String deserializedName = null;

int deserializedAge = 0;

double deserializedSalary = 0.0;

try {

FileInputStream fileIn = new FileInputStream("employee.ser");

ObjectInputStream in = new ObjectInputStream(fileIn);

// Read selected attributes

deserializedName = (String) in.readObject();

deserializedAge = (int) ((Integer) in.readObject()); // Cast to Integer first

deserializedSalary = (double) ((Double) in.readObject()); // Cast to Double first

in.close();

fileIn.close();

// Display deserialized employee

System.out.println("Deserialized Employee:");

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

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

System.out.println("Salary: " + deserializedSalary);

} catch (IOException ioe) {

ioe.printStackTrace();

} catch (ClassNotFoundException cnfe) {

cnfe.printStackTrace();

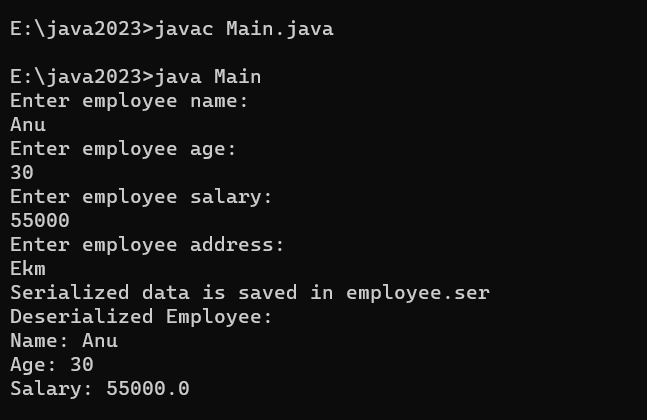
}

scanner.close();

}

}

**Output**



**SECTION 13**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to implement various methods of a StringBuffer class.** |

public class StringBufferExample {

public static void main(String[] args) {

StringBuffer sb = new StringBuffer("Hello"); // Create a StringBuffer object

sb.append(" World"); // Append method

System.*out*.println("After append(): " + sb);

sb.insert(5, " Java"); // Insert method

System.*out*.println("After insert(): " + sb);

sb.delete(5, 10); // Delete method

System.*out*.println("After delete(): " + sb);

sb.reverse(); // Reverse method

System.*out*.println("After reverse(): " + sb);

int length = sb.length(); // Length method

System.*out*.println("Length of StringBuffer: " + length);

int capacity = sb.capacity(); // Capacity method

System.*out*.println("Capacity of StringBuffer: " + capacity);

sb.setLength(5); // SetLength method

System.*out*.println("After setLength(): " + sb);

sb.replace(0, 5, "Hola"); // Replace method

System.*out*.println("After replace(): " + sb);

sb.ensureCapacity(50); // EnsureCapacity method

System.*out*.println("After ensureCapacity(): " + sb);

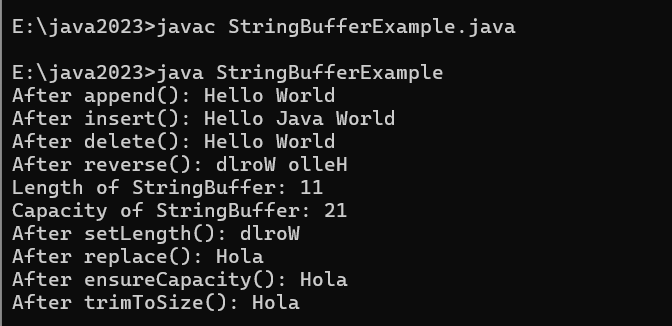
sb.trimToSize(); // TrimToSize method

System.*out*.println("After trimToSize(): " + sb);

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to implement communication between a client and server client, via Socket Programming.** |

package clientserver;

import java.io.\*;

import java.net.\*;

public class Server

{

 public static void main(String[] args)

 {

     try

     {

                ServerSocket ss= new ServerSocket(12345);

                System.out.println("Server is listening...");

                Socket s=ss.accept();

                System.out.println("Made connection with client...");

                ObjectInputStream dis=new ObjectInputStream(s.getInputStream());

                String str=(String)dis.readUTF();

                System.out.println("Message from client is: "+str);

                ss.close();

                dis.close();

     }

     catch (IOException e)

     {

         e.printStackTrace();

     }

 }

}

package clientserver;

import java.io.\*;

import java.net.\*;

public class Client

{

 public static void main(String[] args)

 {

     try

     {

                Socket s=new Socket("localhost",12345);

                System.out.println("Sending a request to client.");

                ObjectOutputStream dout=new ObjectOutputStream(s.getOutputStream());

                dout.writeUTF("Hello Java");

                dout.close();

     }

     catch (IOException e)

     {

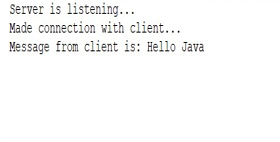
         e.printStackTrace();

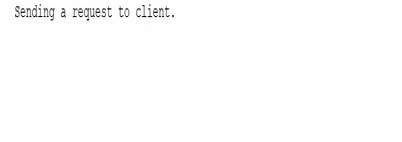
     }

  }

}

**Output**





|  |
| --- |
| **Program 3 Date:** |
| **Write a program to implement communication between a client and server client, via Socket Programming.** |

package javalab;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class server {

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(23561);

System.*out*.println("Server started, waiting for clients...");

Socket socket = serverSocket.accept();

System.*out*.println("Client connected: " + socket);

BufferedReader inputReader = new BufferedReader(newInputStreamReader(socket.getInputStream()));

PrintWriter outputWriter = new PrintWriter(socket.getOutputStream(), true);

BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.*in*));

String receivedMessage, sendMessage;

while (true) {

receivedMessage = inputReader.readLine();

if (receivedMessage.equals("exit")) {

System.*out*.println("Client disconnected");

break;

}

System.*out*.println("Client: " + receivedMessage);

System.*out*.print("You: ");

sendMessage = consoleReader.readLine();

outputWriter.println(sendMessage);

}

socket.close();

serverSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

package javalab;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

public class client {

public static void main(String[] args) {

try {

Socket socket = new Socket("localhost", 23561);

System.out.println("Connected to server: " + socket);

BufferedReader inputReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter outputWriter = new PrintWriter(socket.getOutputStream(), true);

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

String receivedMessage, sendMessage;

while (true) {

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

sendMessage = consoleReader.readLine();

outputWriter.println(sendMessage);

if (sendMessage.equals("exit")) {

System.out.println("Disconnected from server");

break;

}

receivedMessage = inputReader.readLine();

if (receivedMessage.equals("exit")) {

System.out.println("Server disconnected");

break;

}

System.out.println("Server: " + receivedMessage);

}

socket.close();

} catch (IOException e) {

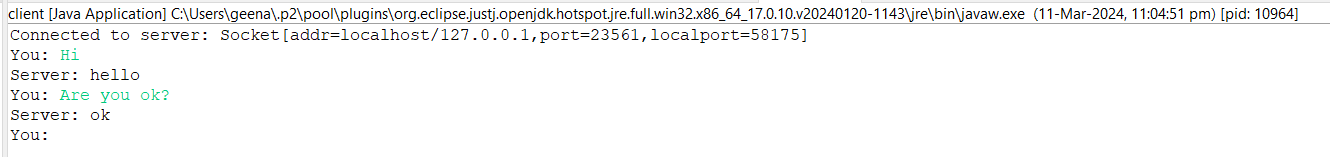
e.printStackTrace();

}

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **Write a program to accept a file name given in the client, and to download the content from the server side.** |

package javalab;

import java.io.\*;

import java.net.\*;

public class Sdemo {

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(8520);

System.*out*.println("Server started, waiting for clients...");

Socket socket = serverSocket.accept();

System.*out*.println("Client connected: " + socket);

BufferedReader inputReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter outputWriter = new PrintWriter(socket.getOutputStream(), true);

String fileName = inputReader.readLine();

System.*out*.println("Client requested file: " + fileName);

File file = new File(fileName);

if (file.exists()) {

BufferedReader fileReader = new BufferedReader(new FileReader(file));

String line;

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

outputWriter.println(line);

}

fileReader.close();

System.*out*.println("File sent successfully");

} else {

outputWriter.println("File not found");

System.*out*.println("File not found");

}

socket.close();

serverSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

package javalab;

import java.io.\*;

import java.net.\*;

public class Cdemo {

public static void main(String[] args) {

try {

Socket socket = new Socket("localhost", 8520);

System.*out*.println("Connected to server: " + socket);

BufferedReader inputReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter outputWriter = new PrintWriter(socket.getOutputStream(), true);

BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.*in*));

System.*out*.print("Enter file name to download: ");

String fileName = consoleReader.readLine();

outputWriter.println(fileName);

String line;

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

if (line.equals("File not found")) {

System.*out*.println("File not found on server");

break;

}

System.*out*.println("Received: " + line);

}

socket.close();

} catch (IOException e) {

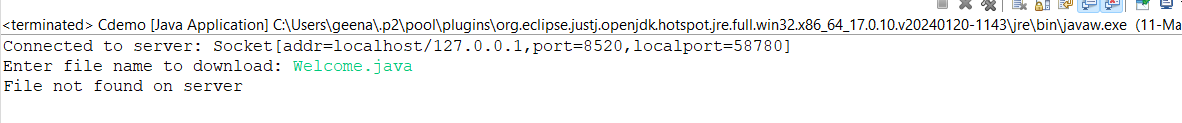
e.printStackTrace();

}

}

}

**Output**



**SECTION 14**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to implement public chatting.** |

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class GroupChat

{

private static final String TERMINATE = "Exit";

static String name;

static volatile boolean finished = false;

public static void main(String[] args)

{

if (args.length != 2)//(239.0.0.0 1234)

System.out.println("Two arguments required: <multicast-host> <portnumber>");

else

{

try

{

InetAddress group = InetAddress.getByName(args[0]);

int port = Integer.parseInt(args[1]);

Scanner sc = new Scanner(System.in);

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

name = sc.nextLine();

MulticastSocket socket = new MulticastSocket(port);

socket.setTimeToLive(0);

socket.joinGroup(group);

Thread t = new Thread(new ReadThread(socket,group,port));

t.start();

System.out.println("Start typing messages...\n");

while(true)

{

String message;

message = sc.nextLine();

if(message.equalsIgnoreCase(GroupChat.TERMINATE))

{

finished = true;

socket.leaveGroup(group);

socket.close();

break;

}

message = name + ": " + message;

byte[] buffer = message.getBytes();

DatagramPacket datagram = new

DatagramPacket(buffer,buffer.length,group,port);

socket.send(datagram);

}

}

catch(SocketException se)

{

System.out.println("Error creating socket");

se.printStackTrace();

}

catch(IOException ie)

{

System.out.println("Error reading/writing from/to socket");

ie.printStackTrace();

}

}

}

}

class ReadThread implements Runnable

{

private MulticastSocket socket;

private InetAddress group;

private int port;

private static final int MAX\_LEN = 1000;

ReadThread(MulticastSocket socket,InetAddress group,int port)

{

this.socket = socket;

this.group = group;

this.port = port;

}

@Override

public void run()

{

while(!GroupChat.finished)

{

byte[] buffer = new byte[ReadThread.MAX\_LEN];//

DatagramPacket datagram = new

DatagramPacket(buffer,buffer.length,group,port);

String message;

try

{

socket.receive(datagram);

message = new String(buffer,0,datagram.getLength(),"UTF8");//parameters?

//if(!message.startsWith(GroupChat.name))//?

System.out.println(message);

}

catch(IOException e)

{

System.out.println("Socket closed!");

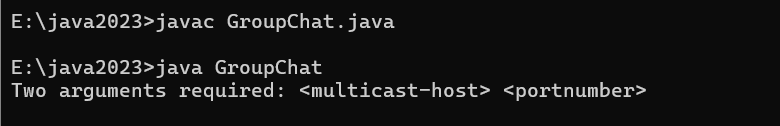
}

}

}

}

**Output**

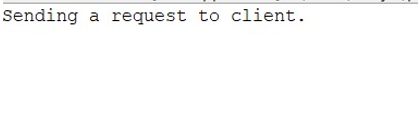
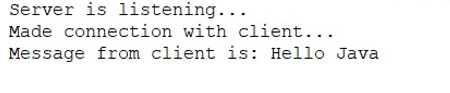


|  |
| --- |
| **Program 2 Date:** |
| **Accept a number from the client, and print its factorial in the server.** |

import java.io.\*;  
import java.net.\*;  
public class SockSer {  
    public static void main(String[] args) {  
        try {  
            ServerSocket ss = new ServerSocket(12345);  
            System.out.println("Waiting patiently for connection from client");  
  
            Socket s = ss.accept(); // Wait for client to connect  
            System.out.println("Made connection with client");  
  
            BufferedReader reader = new BufferedReader(new InputStreamReader(s.getInputStream()));  
            String input = reader.readLine(); // Read number from client  
  
            int number = Integer.parseInt(input);  
            long factorial = calculateFactorial(number);  
  
            PrintWriter writer = new PrintWriter(s.getOutputStream(), true);  
            writer.println(factorial); // Send factorial to client  
            System.out.println("Factorial sent to client: " + factorial);  
            reader.close();  
            writer.close();  
            s.close();  
            ss.close();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
    private static long calculateFactorial(int number) {  
        if (number == 0 || number == 1) {  
            return 1;  
        }  
        long factorial = 1;  
        for (int i = 2; i <= number; i++) {  
            factorial \*= i;  
        }  
        return factorial;  
    }  
}

import java.io.\*;  
import java.net.\*;  
  
public class SockCli {  
    public static void main(String[] args) {  
        try {  
            Socket s = new Socket("localhost", 12345);  
            System.out.println("Sending a request to server");  
  
            BufferedReader userInputReader = new BufferedReader(new InputStreamReader(System.in));  
            System.out.print("Enter a number to calculate its factorial: ");  
            String userInput = userInputReader.readLine();  
  
            PrintWriter writer = new PrintWriter(s.getOutputStream(), true);  
            writer.println(userInput); // Send number to server  
  
            BufferedReader reader = new BufferedReader(new InputStreamReader(s.getInputStream()));  
            String response = reader.readLine(); // Receive factorial from server  
  
            System.out.println("Factorial received from server: " + response);  
  
            userInputReader.close();  
            writer.close();  
            reader.close();  
            s.close();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}

**Output**

**SECTION 15**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to implement connection less communication.** |

package javalab;

import java.net.\*;

public class SerDemo{

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

DatagramSocket ds = new DatagramSocket();

String str = "Welcome java";

InetAddress ip = InetAddress.*getByName*("127.0.0.1");

DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);

ds.send(dp);

ds.close();

}

}

package javalab;

import java.net.\*;

public class CliDemo{

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

DatagramSocket ds = new DatagramSocket(3000);

byte[] buf = new byte[1024];

DatagramPacket dp = new DatagramPacket(buf, 1024);

ds.receive(dp);

String str = new String(dp.getData(), 0, dp.getLength());

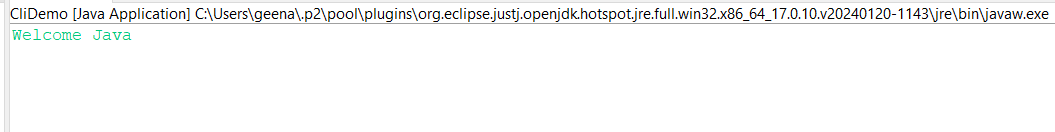
System.*out*.println(str);

ds.close();

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program to get protocol, file name, host, path and port of a given URL.** |

import java.net.\*;

public class urldemo{

public static void main(String[] args){

try{

URL url=new URL("http://www.javatpoint.com/java-tutorial");

System.out.println("Protocol: "+url.getProtocol());

System.out.println("Host Name: "+url.getHost());

System.out.println("Port Number: "+url.getPort());

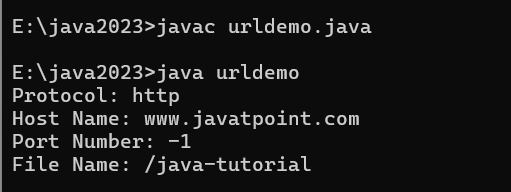
System.out.println("File Name: "+url.getFile());

}catch(Exception e){System.out.println(e);}

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a program to download a file from a given URL** |

package javalab;

import java.io.\*;

import java.net.\*;

public class contentdemo

{

public static void main(String[] args)

{

try {

URL url = new URL("https://www.wikipedia.org");

URLConnection urlConnection = url.openConnection();

BufferedReader in = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()));

String inputLine;

System.*out*.println("Content of the URL:");

while ((inputLine = in.readLine()) != null)

{

System.*out*.println(inputLine);

}

in.close();

}

catch (IOException e)

{

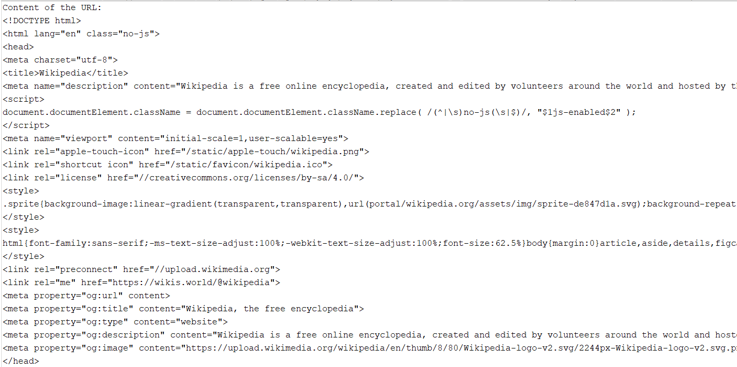
e.printStackTrace();

}

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **Implement Two- way Communication using UDP Protocol.** |

package javalab;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class udpServer {

public static void main(String[] args) {

DatagramSocket socket = null;

try {

// Create a server socket bound to port 9876

socket = new DatagramSocket(9872);

System.*out*.println("Server started...");

// Receive messages from clients indefinitely

while (true) {

byte[] receiveData = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

socket.receive(receivePacket);

// Convert byte array to string and print the received message

String receivedMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());

System.*out*.println("Client: " + receivedMessage);

}

} catch (IOException e) {

e.printStackTrace();

} finally {

if (socket != null) {

socket.close();

}

}

}

}

package javalab;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class udpCli {

public static void main(String[] args) {

DatagramSocket socket = null;

try {

// Create a client socket

socket = new DatagramSocket();

// Read input from the user

BufferedReader userInput = new BufferedReader(new InputStreamReader(System.*in*));

while (true) {

// Read message from the user

System.*out*.print("You: ");

String message = userInput.readLine();

// Convert message to byte array

byte[] sendData = message.getBytes();

// Specify server address and port

InetAddress serverAddress = InetAddress.*getByName*("localhost");

int serverPort = 9872;

// Create packet with message and send to server

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddress, serverPort);

socket.send(sendPacket);

}

} catch (IOException e) {

e.printStackTrace();

} finally {

if (socket != null) {

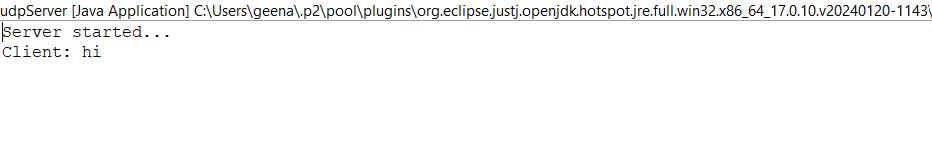
socket.close();

}

}

}}

**Output**



**SECTION 16**

|  |
| --- |
| **Program 1 Date:** |
| **Write a program to create a table Person ( P\_Id(Primary), P\_Name, P\_age, P\_address, P\_DOB), insert records, and display the records.** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class person {

public static void main(String args[]) {

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

Statement stmt = con.createStatement();

stmt.executeUpdate("create table person(pid varchar2(20) primary key,pname varchar2(20),page number,paddress varchar2(20),pdob number)");

System.out.println("Table created...");

con.close();

} catch (ClassNotFoundException | SQLException e) {

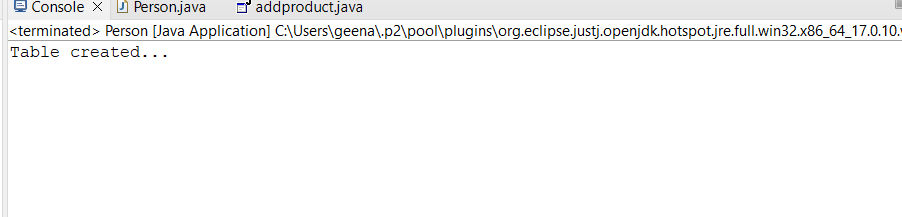
System.out.println(e);

}

}

}

**Output**



package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class InsPerson {

public static void main(String args[]) {

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:ORCL", "system", "password”);

Statement stmt = con.createStatement();

stmt.executeUpdate("insert into person values('P1',’Geena’,22,’EKM’,2001)");

System.out.println("Value inserted...");

con.close();

} catch (ClassNotFoundException | SQLException e) {

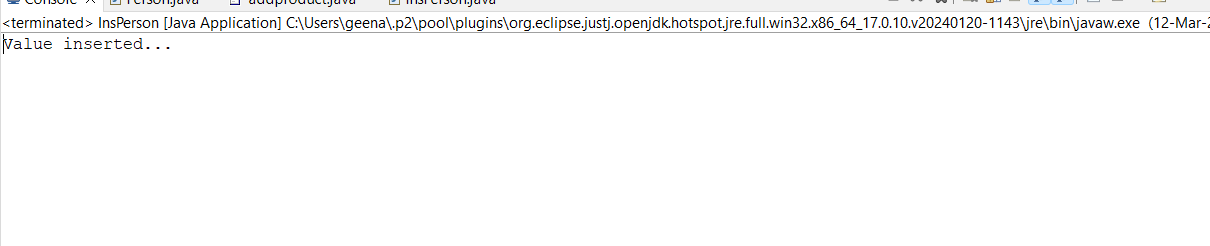
System.out.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Assume that Login is a table which has Uname, Upass. Check whether a record with “Uname=”Alexa” and “UPass=”Siri123#”is present in the table.** |

package javalab;

import java.sql.\*;

public class Login {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

String username = "Alexa";

String password = "Siri123#";

String query = "SELECT COUNT(\*) FROM login WHERE uname = '" + username + "' AND password = '" + password + "'";

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery(query);

rs.next();

int count = rs.getInt(1);

if (count > 0) {

System.*out*.println("Record found for username 'Alexa' and password 'Siri123#'");

} else {

System.*out*.println("No record found for username 'Alexa' and password 'Siri123#'");

}

rs.close();

stmt.close();

con.close();

} catch (ClassNotFoundException | SQLException e) {

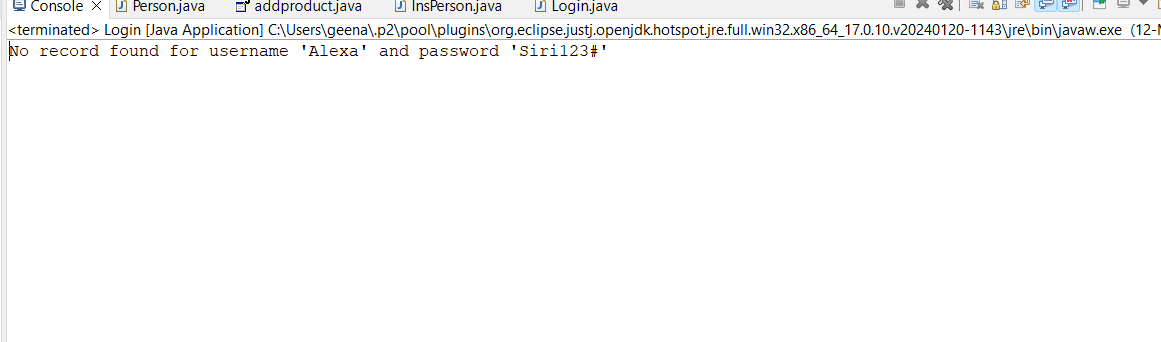
System.*out*.println(e);

}

}

}

**Output**



**SECTION 17**

|  |
| --- |
| **Program 1 Date:** |
| **Construct the following tables:**  **Department (dno(Primary), dname, dloc)**  **Emp ( eno(Primary), ename, esal ,dno(Foreign))** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

//import java.sql.SQLException;

import java.sql.Statement;

public class Dept {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL","system","password");

Statement stmt = con.createStatement();

stmt.executeUpdate("create table Departments(dno varchar2(20) primary key,dname varchar2(20),dloc varchar2(20))");

System.*out*.println("Table created...");

con.close();

} catch (Exception e) {

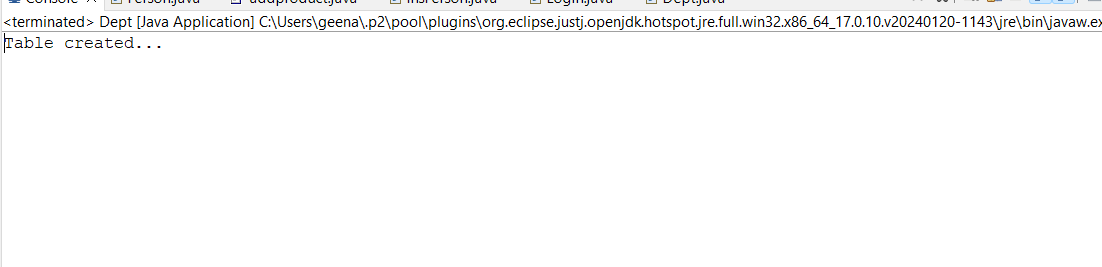
System.*out*.println(e);

}

}

}

**Output**



package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

//import java.sql.SQLException;

import java.sql.Statement;

public class Emp {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL","system","password");

Statement stmt = con.createStatement();

stmt.executeUpdate("create table Employees(emp varchar2(20) primary key,ename varchar2(20),esal varchar2(20),dno varchar2(20) references Departments(dno))");

System.*out*.println("Table created...");

con.close();

} catch (Exception e) {

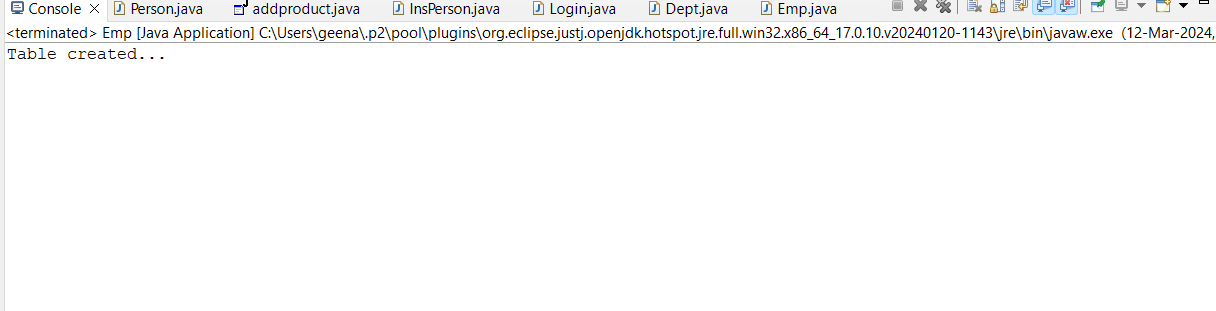
System.*out*.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a program for displaying information in the following order from the above tables:**  **eno          ename                     esal         dname           dloc**  **101                Chetan               10,000        Civil      Kochi**  **102              Amish                  20,000     Accounts      Delhi** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Statement;

public class DisplayInfo {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT e.emp, e.ename, e.esal, d.dname, d.dloc FROM employees e, departments d WHERE e.dno = d.dno");

ResultSetMetaData md = rs.getMetaData();

for (int i = 1; i <= md.getColumnCount(); i++) {

System.*out*.print(" " + md.getColumnName(i));

}

System.*out*.println("\n");

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

int rowCount = 0;

while (rs.next()) {

rowCount++;

for (int i = 1; i <= md.getColumnCount(); i++) {

System.*out*.print(rs.getString(i) + "\t");

}

System.*out*.println("\n");

}

con.close();

} catch (ClassNotFoundException | SQLException e) {

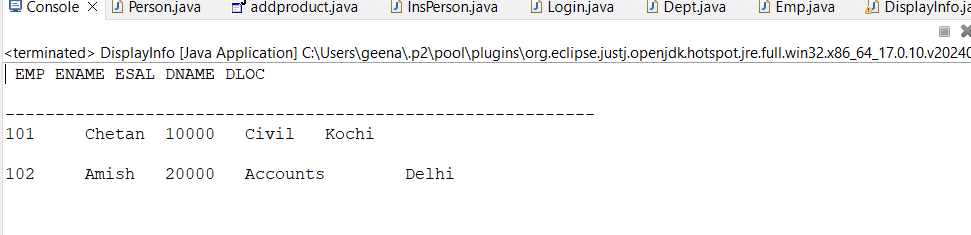
System.*out*.println(e);

}

}

}

**Output**



**SECTION 18**

|  |
| --- |
| **Program 1 Date:** |
| **Program to implement database connectivity using object oriented concepts.** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Statement;

public class oocConcept {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from person");

while(rs.next()) {

System.*out*.println(rs.getString(1));

System.*out*.println(rs.getString(2));

System.*out*.println(rs.getInt(3));

System.*out*.println(rs.getString(4));

System.*out*.println(rs.getInt(5));

}

con.close();

} catch (ClassNotFoundException | SQLException e) {

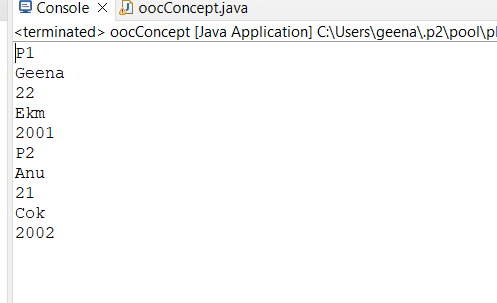
System.*out*.println(e);

}

}

}

**Output**



**SECTION 19**

|  |
| --- |
| **Program 1 Date:** |
| **Write a JDBC program with Parametrized queries to update a given record (Rani’s salary to 15,000) in the Emp table.** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class UDemo {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

Statement stmt = con.createStatement();

PreparedStatement pst=con.prepareStatement("update employees set esal=? where emp=?");

pst.setInt(1,15000);//1 specifies the first parameter in the query i.e. name

pst.setString(2,"E102");

int i=pst.executeUpdate();

System.*out*.println(i+" records updated");

con.close();

} catch (ClassNotFoundException | SQLException e) {

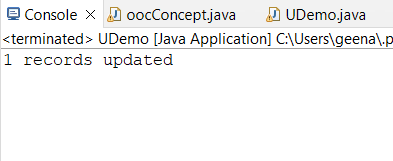
System.*out*.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Write a JDBC program with Parametrized queries to list the records of Emp table which has records whose names start with the alphabet “R”.** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class ParamSel {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

Statement stmt = con.createStatement();

PreparedStatement pst = con.prepareStatement("select \* from employees where ename like 'R%'");

ResultSet rs = pst.executeQuery();

while (rs.next()) {

System.*out*.println(rs.getString(1) + " " + rs.getString(2) + " " + rs.getInt(3) + " " + rs.getString(4));

}

con.close();

} catch (ClassNotFoundException | SQLException e) {

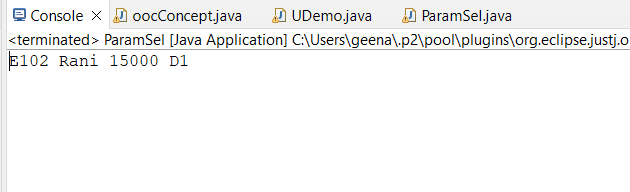
System.*out*.println(e);

}

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a JDBC program with PreparedStatement to delete the records of Emp table which has records whose salary is less than 10,000.** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class ParamDel {

public static void main(String args[]) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

Statement stmt = con.createStatement();

PreparedStatement pst=con.prepareStatement("delete from employees where esal<?");

pst.setInt(1,10000);

int i=pst.executeUpdate();

System.*out*.println(i+" records deleted");

con.close();

} catch (ClassNotFoundException | SQLException e) {

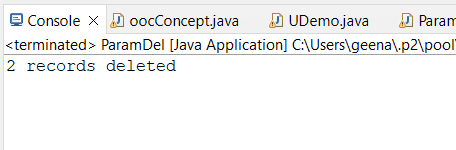
System.*out*.println(e);

}

}

}

**Output**



**SECTION 20**

|  |
| --- |
| **Program 1 Date:** |
| **Implement a JDBC program which uses a Stored Procedure to insert records into Department table.** |

package javalab;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.DriverManager;

public class StoredProcedureIns {

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

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

CallableStatement stmt=con.prepareCall("{call insertR(?,?,?)}");

stmt.setString(1,"d03");

stmt.setString(2,"cs");

stmt.setString(3,"clct");

stmt.execute();

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

}

}

**Output**



|  |
| --- |
| **Program 2 Date:** |
| **Use Callable statement to implement a Stored Procedure to display the Ename and Salary of all employees.** |

package javalab;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import oracle.jdbc.OracleTypes;

public class DispEmp {

public static void main(String[] args) {

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

CallableStatement cstmt = con.prepareCall("{call displayEmployees(?)}");

cstmt.registerOutParameter(1, OracleTypes.*CURSOR*);

cstmt.execute();

ResultSet rs = (ResultSet) cstmt.getObject(1);

while (rs.next()) {

String ename = rs.getString("ename");

double salary = rs.getDouble("esal");

System.*out*.println("Employee Name: " + ename + ", Salary: " + salary);

}

System.*out*.println("Stored procedure executed successfully.");

} catch (SQLException | ClassNotFoundException e) {

System.*err*.println("Error executing stored procedure: " + e.getMessage());

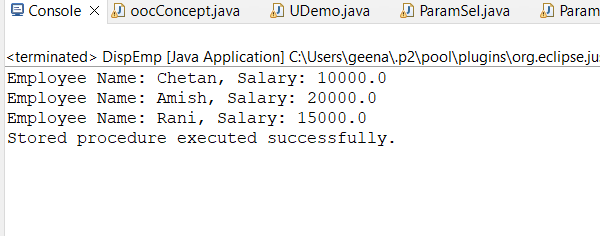
e.printStackTrace();

}

}

}

**Output**



|  |
| --- |
| **Program 3 Date:** |
| **Write a JDBC program to implement Transaction Management in the Department table.** |

package javalab;

import java.sql.Statement;

import java.sql.Connection;

import java.sql.DriverManager;

public class TransactionDemo {

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

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:ORCL", "system", "password");

con.setAutoCommit(false);

Statement stmt=con.createStatement();

stmt.executeUpdate("insert into departments values('d04','mlylm','ernklm')");

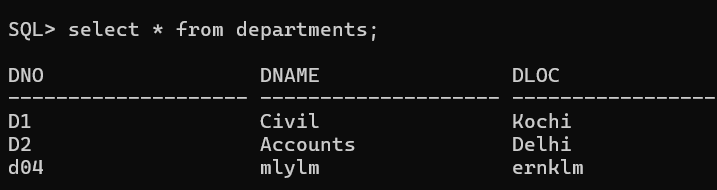
con.commit();

con.close();

}

}

**Output**



|  |
| --- |
| **Program 4 Date:** |
| **Write a JDBC program to depict the usage of SQLException Class and SQLWarning Class** |

package javalab;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class SqlExcp{

public static void main(String[] args) {

Connection connection = null;

Statement statement = null;

ResultSet resultSet = null;

try {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection connection1 = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:ORCL", "hr", "1234rh#");

Statement statement1 = connection1.createStatement();

ResultSet resultSet1 = statement1.executeQuery("SELECT \* FROM flower");

// Processing the result set

while (resultSet1.next()) {

// Code to process each row of the result set

}

} catch (ClassNotFoundException e) {

System.*err*.println("Failed to load Oracle JDBC driver");

e.printStackTrace();

} catch (SQLException e) {

// Handling SQLException

System.*err*.println("SQLException occurred:");

while (e != null) {

System.*err*.println("SQLState: " + e.getSQLState());

System.*err*.println("Error Code: " + e.getErrorCode());

System.*err*.println("Message: " + e.getMessage());

e = e.getNextException();

}

} finally {

// Closing resources in finally block

try {

if (resultSet != null) {

resultSet.close();

}

if (statement != null) {

statement.close();

}

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

// Handling SQLException while closing resources

e.printStackTrace();

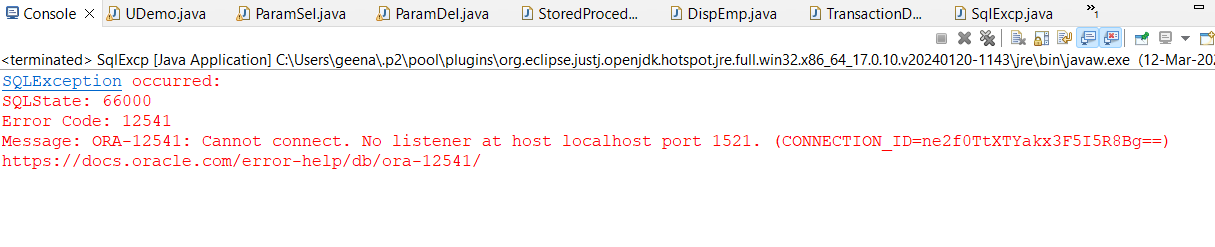
}

}

}

}

**Output**

****