**JAVA**  **LABCYCLE-1**

**Experiment 1**

**Aim:** Java program to accept mark of a student into a 1D array and find total mark and percentage.

**Algorithm:**

1. Start
2. Declare an array int a[].
3. Declare the variable and initialize total=0 ,i,n, count = 0, percentage.
4. Ask the user to enter the no of subject that stored to –> n and Enter the mark initialize to the array a[].
5. calculate the total and percentage of all the marks in an array.
6. Update the total in each iteration. for(int i = 0; i < n; i++)

a[i] = s.nextInt();total = total + a[i]; percentage = total / n;

1. Print the total.
2. Print the percentage secured.
3. Stop.

**Program:**

import java.util.Scanner;

class Stud\_Marks

{

public static void main(String[] args)

{

int n, total = 0, p;

Scanner s = new Scanner(System.in);

System.out.print("Enter no. of subject:");

n = s.nextInt();

int marks[] = new int[n];

System.out.println("Enter marks out of 100:");

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

{

marks[i] = s.nextInt();

total = total + marks[i];

}

p = total / n;

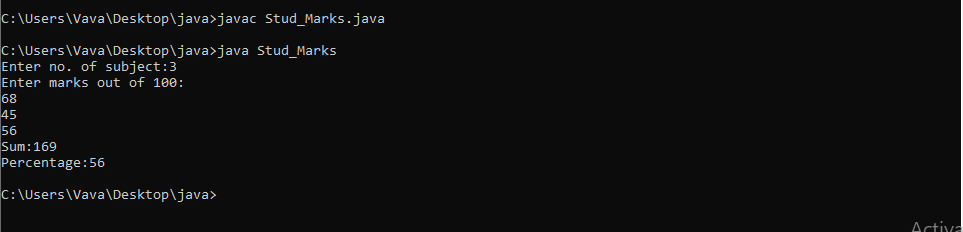
System.out.println("Sum:"+total);

System.out.println("Percentage:"+p);

}

}

**Output:**

****

**Experiment 2**

**Aim:**java program to count the number of occurrence of an element in an array

**Algorithm:**

1. Start
2. Declare an array, initialize n, x, count = 0, i = 0
3. Ask the user to initialize the array
4. enter all elements in the array
5. for(i=0;i<arr.length;i++) : a[i] store the elements In the array
6. Ask the user to initialize count number of occurrences
7. x = s.nextInt() store the count
8. for(i = 0; i < n; i++) if(a[i] == x)
9. count++;
10. print Number of Occurrence of the Element:"+count
11. Stop

**Program:**

import java.util.Scanner;

public class CountOccurrence

{

public static void main(String[] args)

{

int n, item, count = 0;

Scanner s = new Scanner(System.in);

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

n = s.nextInt();

int a[] = new int[n];

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

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

{

a[i] = s.nextInt();

}

System.out.println("Enter the element to be searched:");

item = s.nextInt();

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

{

if(a[i] == item)

{

count++;

}

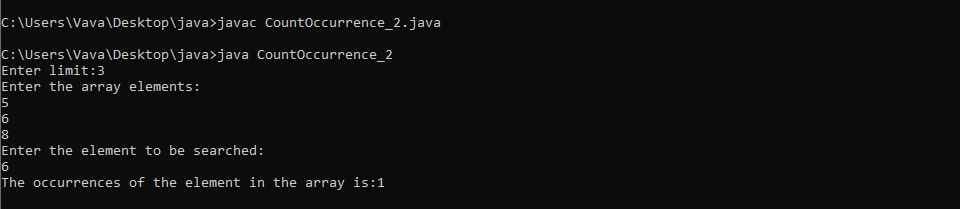
}

System.out.println("The occurrences of the element in the array is:"+count);

}

}

**Output:**

****

**Experiment 3**

**Aim:** Read two matrices from 5the console and perform matrix addition

**Algorithm:**

1. If both matrices are of the same size then only we can add the matrices.

2. Use the double dimensional array to store the matrix elements.

3. Read row number, column number and initialize the double dimensional arrays a[][],b[][],c[][] with same row number, column number.

4. Store the first matrix elements into the two-dimensional array a[][] using two for loops. i indicates row number, j indicates column index. Similarly

Matrix 2 elements in to b[][].

5. Add the two matrices using for loop

for i=0 to i<row

for j=0 to j<col

a[i][j] + b[i][j] and store it in to the matrix res at c[i][j] .

**Program:**

import java.util.Scanner;

public class Add\_Matrix

{

public static void main(String[] args)

{

int p, q, m, n;

Scanner s = new Scanner(System.in);

System.out.print("Enter number of rows in first matrix:");

p = s.nextInt();

System.out.print("Enter number of columns in first matrix:");

q = s.nextInt();

System.out.print("Enter number of rows in second matrix:");

m = s.nextInt();

System.out.print("Enter number of columns in second matrix:");

n = s.nextInt();

if (p == m && q == n)

{

int a[][] = new int[p][q];

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

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

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

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

{

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

{

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

}

}

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

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

{

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

{

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

}

}

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

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

{

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

{

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

}

System.out.println("");

}

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

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

{

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

{

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

}

System.out.println("");

}

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

{

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

{

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

{

c[i][j] = a[i][j] + b[i][j];

}

}

}

System.out.println("Matrix after addition:");

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

{

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

{

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

}

System.out.println("");

}

}

else

{

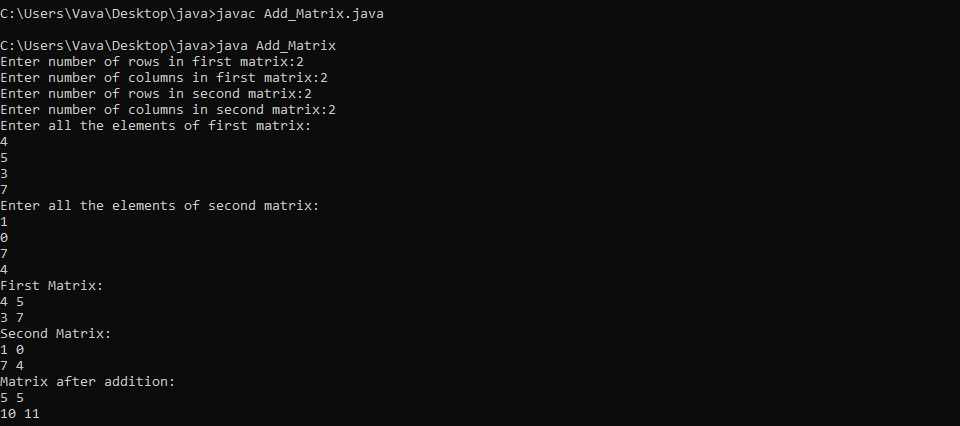
System.out.println("Addition would not be possible");

}

}

}

**Output:**

****

**Experiment 4**

**Aim:** Add complex numbers

**Algorithm:**

1. Define class Complex Number with members real, img of double datatype

for the real and imaginary part of the complex number.

2. Define a constructor to initialise real and img.

3. Create a temporary complex number to hold the sum of two numbers

4. Do the summation and return the value.

**Program:**

import java.util.Scanner;

class ComplexNumber{

double real,img;

ComplexNumber(double r,double i){

this.real = r;

this.img = i;

}

public static ComplexNumber sum(ComplexNumber c1, ComplexNumber c2){

ComplexNumber temp = new ComplexNumber(0,0);

temp.real = c1.real + c2.real;

temp.img = c1.img + c2.img;

return temp;

}

public static void main(String args[]){

Scanner input = new Scanner(System.in);

System.out.println("Enter real part of first number:");

float n1 = input.nextFloat();

System.out.println("Enter imaginary part of first number:");

float n2 = input.nextFloat();

System.out.println("Enter real part of Second number:");

float n3 = input.nextFloat();

System.out.println("Enter imaginary part of Second number:");

float n4 = input.nextFloat();

ComplexNumber c1 = new ComplexNumber(n1,n2);

ComplexNumber c2 = new ComplexNumber(n3,n4);

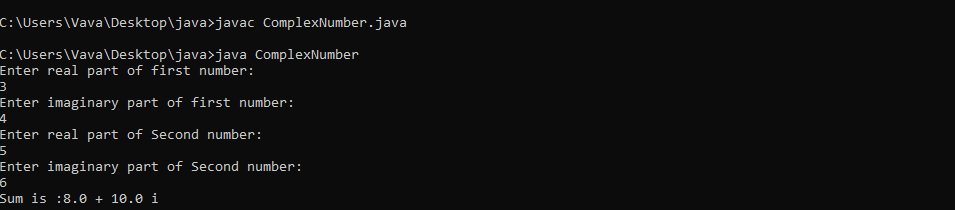
ComplexNumber temp = sum(c1,c2);

System.out.println("Sum is :"+temp.real+ " + "+temp.img+" i" );

}

}

**Output:**

****

**Experiment 5**

**Aim:** Read a matrix from the console and check whether it is symmetric or not

**Algorithm:**

1. Start
2. declare array matrix[][]
3. Read row number, column number and initialize rows,cols
4. Store the matrix elements into the two-dimensional array matrix[i][j] using two for loops. i indicates row number, j indicates column index.
5. if(rows != cols) then The given matrix is not a square matrix, so it can't be symmetric.
6. Find transpose of the matrix
7. Compare two matrices.
8. If the two matrices is the same then it is symmetric otherwise it's not.
9. Stop

**Program:**

import java.util.\*;

public class symmetric {

static void checkSymmetric(int mat[][], int row,

int col)

{

int i, j, flag = 1;

System.out.println("The matrix formed is:");

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

for (j = 0; j < col; j++) {

System.out.print(mat[i][j] + "\t");

}

System.out.println("");

}

int[][] transpose = new int[row][col];

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

for (j = 0; j < col; j++) {

transpose[j][i] = mat[i][j];

}

}

if (row == col) {

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

for (j = 0; j < col; j++) {

if (mat[i][j] != transpose[i][j]) {

flag = 0;

break;

}

}

if (flag == 0) {

System.out.print(

"\nThe matrix is not symmetric");

break;

}

}

if (flag == 1) {

System.out.print(

"\nThe matrix is symmetric");

}

}

else {

System.out.print(

"\nThe matrix is not symmetric");

}

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int i, j, row, col, flag = 1;

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

row = sc.nextInt();

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

col = sc.nextInt();

int[][] mat = new int[row][col];

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

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

// Inner loop for colummns

for (j = 0; j < col; j++) {

// Print matrix element

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

}

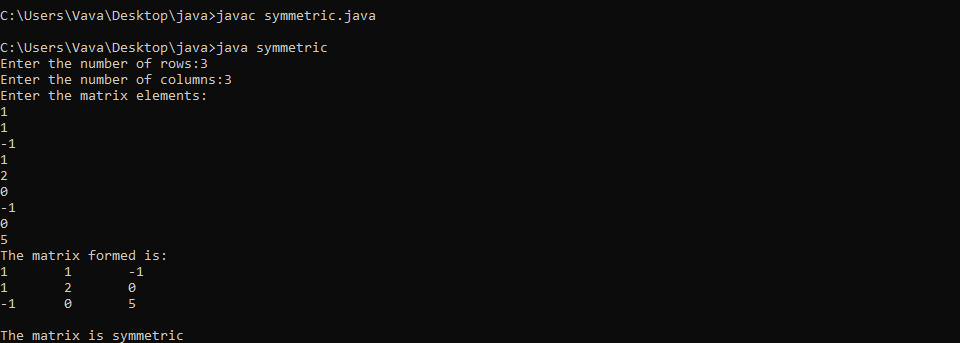
}

checkSymmetric(mat, row, col);

}

}

**Output:**

****

**Experiment 5**

**Aim:** Define a class ‘product’ with data members pcode,pname and price.Create 3 objects of the class and find the product having lowest price

**Algorithm:**

1. start
2. declare the class product
3. initialize data member pcode=0,pname=null,price=0 in default constructor product.
4. declare a member function void cal and compare the lowest price of product.
5. create 3 object in class product
6. Ask the user to enter 3 pcode and pname,price and stored using object of class product
7. performing comparison and print the output.
8. stop

**Program:**

import java.util.\*;

public class Product

{

int pcode;

String pname;

int price;

Product()

{

pcode=0;

pname=null;

price=0;

}

public static void cal(int a,int b,int c)

{

int p1=a;

int p2=b;

int p3=c;

int lowest;

if(p1<p2)

{

if(p3<p1)

{

lowest = p3;

}

else

{

lowest = p1;

}

}

else

{

if(p2<p3)

{

lowest = p2;

}

else

{

lowest = p3;

}

}

System.out.println("The lowest price is "+lowest);

}

public static void main(String args[])

{

Scanner s = new Scanner(System.in);

Product p1= new Product();

Product p2= new Product();

Product p3= new Product();

System.out.print("enter the product1 id : ");

p1.pcode=s.nextInt();

System.out.print("enter the product1 name : ");

p1.pname=s.next();

System.out.print("enter the product1 price : ");

p1.price=s.nextInt();

System.out.print("enter the product2 id : ");

p2.pcode = s.nextInt();

System.out.print("enter the product2 name : ");

p2.pname=s.next();

System.out.print("enter the product2 price : ");

p2.price=s.nextInt();

System.out.print("enter the product3 id : ");

p3.pcode = s.nextInt();

System.out.print("enter the product3 name : ");

p3.pname=s.next();

System.out.print("enter the product3 price : ");

p3.price=s.nextInt();

Product.cal(p1.price,p2.price,p3.price);

}

}

**Output:**

****

**Experiment 7**

**Aim:** Create CPU with attribute price. Create inner class processor (no.o f cores, manufacturer) and static nested class RAM (memory manufacturer).Create an object of CPU and print information of processor and RAM

**Algorithm:**

1. Define\_class CPU with member price of double datatype.

2. Define an inner class Processor with members cores (double) and manu-

facturer (String).

3. Define another nested protected class RAM with members memory and

manufacturer of double and String data types respectively.

4. Define a public class CPUDetails

5. Create object of Outer class CPU i.e., CPU cpu = new CPU();

6. Create an object of inner class Processor using outer class CPU.Processor

processor = cpu.new Processor();

7. Create an object of inner class RAM using outer class CPU CPU.RAM

ram = cpu.new RAM();

8. Print Processor cache= processor.getCache();

9. Print Ram Clock speed = ram.getClockSpeed();

10. End class CPUDetails.

**Program:**

class CPU

{

double price;

class Processor

{

double cores;

String manufacturer;

double getCache()

{

return 4.3 ;

}

}

protected class RAM

{

double memory;

String manufacturer;

double getClockSpeed()

{

return 5.5 ;

}

}

}

public class Main

{

public static void main(String[] args)

{

CPU cpu = new CPU();

CPU.Processor processor = cpu.new Processor();

CPU.RAM ram = cpu.new RAM();

System.out.println ("Processor Cache = " + processor.getCache());

System.out.println ("Ram Clock speed = " + ram. getClockSpeed());

}

}

**Output:**

****