**JAVA LAB CYCLE-1**

## Experiment 1: Use Array to Find Total Marks and Percentage.

**Aim**: Accept the Marks of a Student into a 1D Array and find total marks 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:**

**percentage.java**

import java.util.Scanner;

public class percentage

{

public static void main(String[] args)

{

int n, count = 0, total = 0, percentage;

Scanner s = new Scanner(System.in);

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

n = s.nextInt();

int a[] = new int[n];

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

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

{

a[i] = s.nextInt();

total = total + a[i];

}

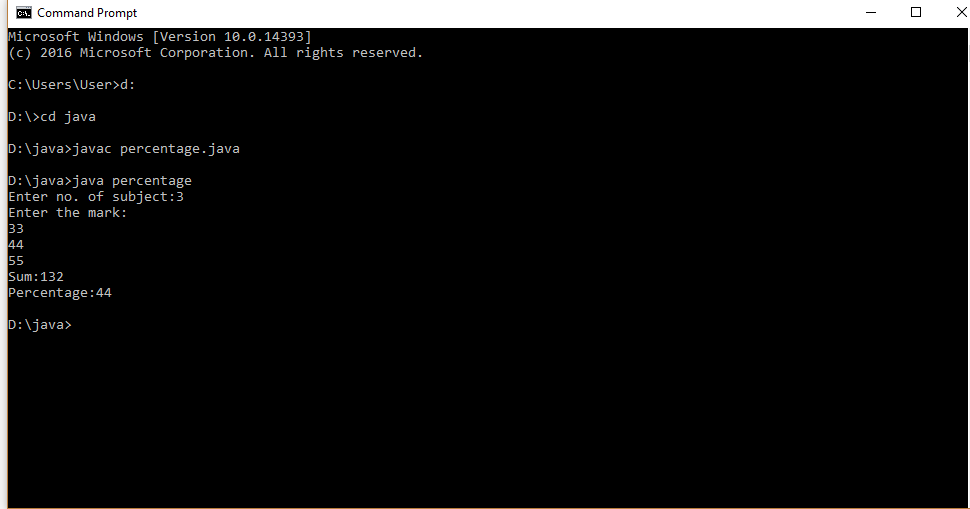
percentage = total / n;

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

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

}

}

**Expected Output** 

## Experiment 2: Count the Number of Occurrence in an Array

**Aim**: 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:**

**CountOccurrence.java**

import java.util.Scanner;

public class CountOccurrence

{

public static void main(String[] args)

{

int n, x, count = 0, i = 0;

Scanner s = new Scanner(System.in);

System.out.print("Enter no. of elements you want in array:");

n = s.nextInt();

int a[] = new int[n];

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

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

{

a[i] = s.nextInt();

}

System.out.print("Enter the element of which you want to count number of occurrences:");

x = s.nextInt();

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

{

if(a[i] == x)

{

count++;

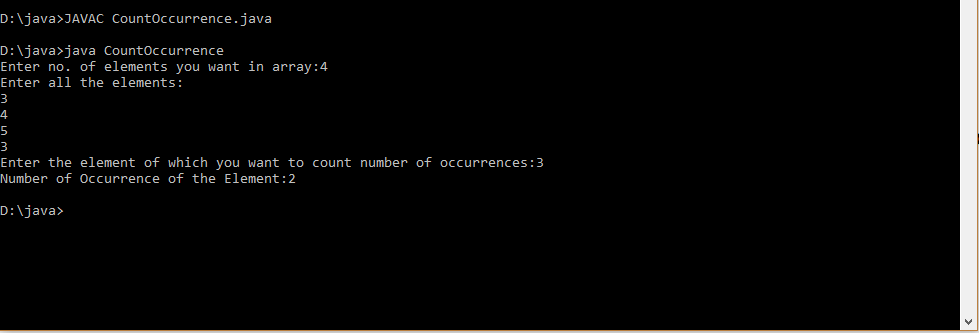
}

}

System.out.println("Number of Occurrence of the Element:"+count);

}

}

**Expected Output** 

## Experiment 3: Matrix Addition

**Aim**: Add Two MXN Matrix from User Input.

**Algorithm**

1. Start
2. declare two   matrices are of the same size.
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

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

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

1. stop.

**Program:**

**multi.java**

import java.util.Scanner;

public class multi

{

public static void main(String[] args)

{

int p, q;

Scanner s = new Scanner(System.in);

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

p = s.nextInt();

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

q = s.nextInt();

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

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

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

System.out.println("Enter all the elements of 1 st matrix:");

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

{

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

{

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

}

}

System.out.println("The Matrix is:");

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("Enter all the elements of 2 nd matrix:");

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

{

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

{

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

}

}

System.out.println("The Matrix is:");

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

{

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

{

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

}

System.out.println("");

}

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

{

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

{

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

}

}

System.out.println("Sum of the matrices:-");

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

{

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

{

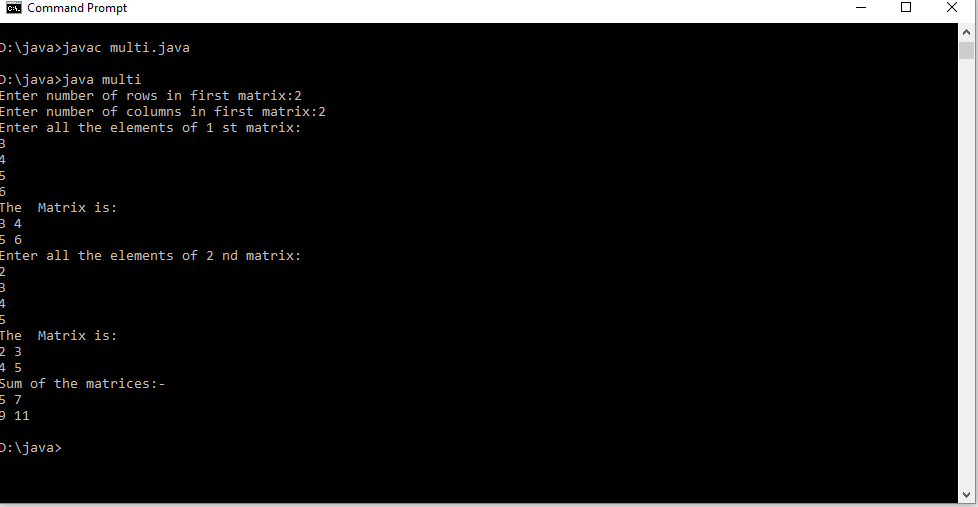
System.out.print(c[i][j]+" ");

}

System.out.println("");

}

}}

**Expected Output** 

## Experiment 4: Add Complex Numbers

**Aim**: Add two complex numbers .

**Algoritham**

1. Start
2. Declare the class complex
3. define the real and imaginary part of a complex number
4. Declare and define function to add the complex numbers(parameterized constructor )
5. Declare the main() class with object
6. Enter complex numbers to be added
7. Call the function to add and perform addition operation
8. Print the Result
9. stop

**Program:**

**GFG.java**

import java.util.\*;

class Complex {

int real, imaginary;

Complex()

{

}

Complex(int tempReal, int tempImaginary)

{

real = tempReal;

imaginary = tempImaginary;

}

Complex addComp(Complex C1, Complex C2)

{

Complex temp = new Complex();

temp.real = C1.real + C2.real;

temp.imaginary = C1.imaginary + C2.imaginary;

return temp;

}

}

public class GFG

{

public static void main(String[] args)

{

Complex C1 = new Complex(3, 2);

System.out.println("Complex number 1 : "

+ C1.real + " + i"

+ C1.imaginary);

Complex C2 = new Complex(9, 5);

System.out.println("Complex number 2 : "

+ C2.real + " + i"

+ C2.imaginary);

Complex C3 = new Complex();

C3 = C3.addComp(C1, C2);

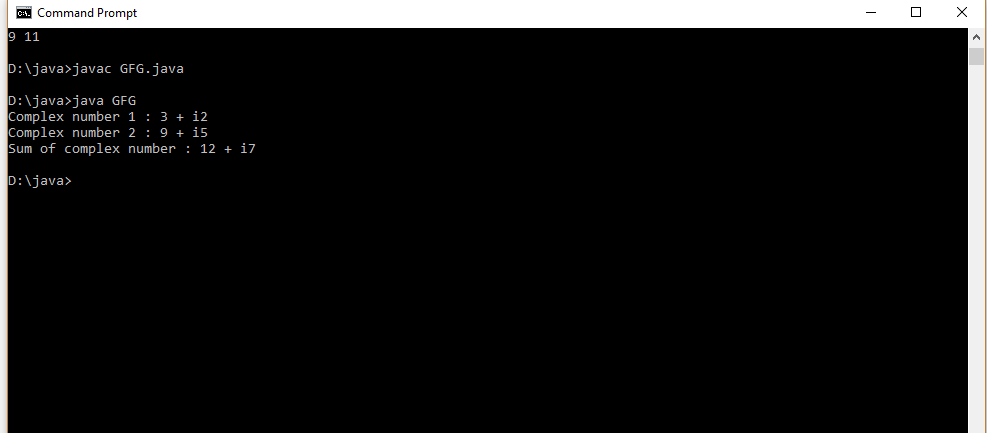
System.out.println("Sum of complex number : "

+ C3.real + " + i"

+ C3.imaginary);

}

}

**Expected Output** 

**Experiment 5: Symmetric matrix**

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

**Algoritham**

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:**

**Symmetric.java**

import java.util.Scanner;

public class Symmetric

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the no. of rows : ");

int rows = sc.nextInt();

System.out.println("Enter the no. of columns : ");

int cols = sc.nextInt();

int matrix[][] = new int[rows][cols];

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

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

{

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

{

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

}

}

System.out.println("Printing the input matrix :");

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

{

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

{

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

}

System.out.println();

}

if(rows != cols)

{

System.out.println("The given matrix is not a square matrix, so it can't be symmetric.");

}

else

{

boolean symmetric = true;

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

{

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

{

if(matrix[i][j] != matrix[j][i])

{

symmetric = false;

break;

}

}

}

if(symmetric)

{

System.out.println("The given matrix is symmetric...");

}

else

{

System.out.println("The given matrix is not symmetric...");

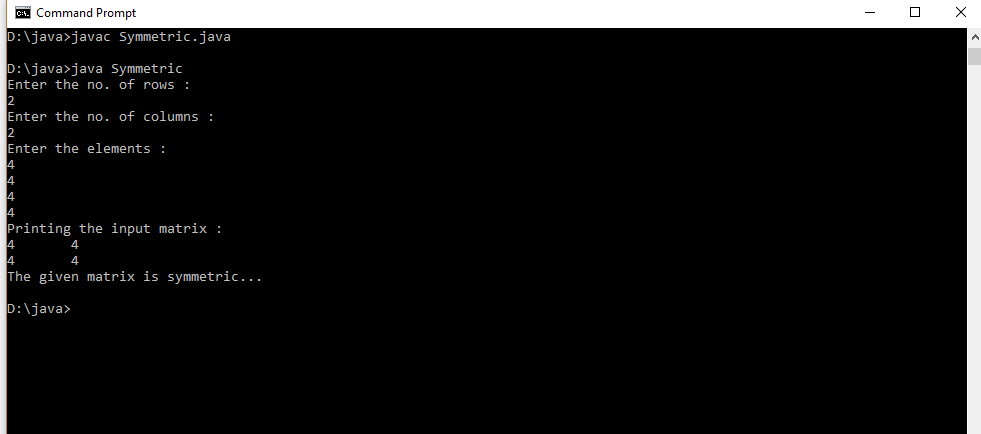
}

}

sc.close();

}

}

**Expected Output** 

## Experiment 6: Design and use ‘Product’ class

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

**Products.java**

import java.util.\*;

public class Products

{

int pcode;

String pname;

int price;

Products()

{

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;

float 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 among the 3 Product is : "+lowest);

}

public static void main(String args[])

{

Scanner s = new Scanner(System.in);

Products p1= new Products();

Products p2= new Products();

Products p3= new Products();

System.out.print("Enter the Product 1 Code : ");

p1.pcode =s.nextInt();

System.out.print("Enter the Product 1 Name : ");

p1.pname = s.next();

System.out.print("Enter the Product 1 Price : ");

p1.price=s.nextInt();

System.out.print("Enter the product 2 Code : ");

p2.pcode =s.nextInt();

System.out.print("Enter the Product 2 Name : ");

p2.pname = s.next();

System.out.print("Enter the Product 2 Price : ");

p2.price = s.nextInt();

System.out.print("Enter the Product 3 Code : ");

p3.pcode =s.nextInt();

System.out.print("Enter the Product 3 Name : ");

p3.pname = s.next();

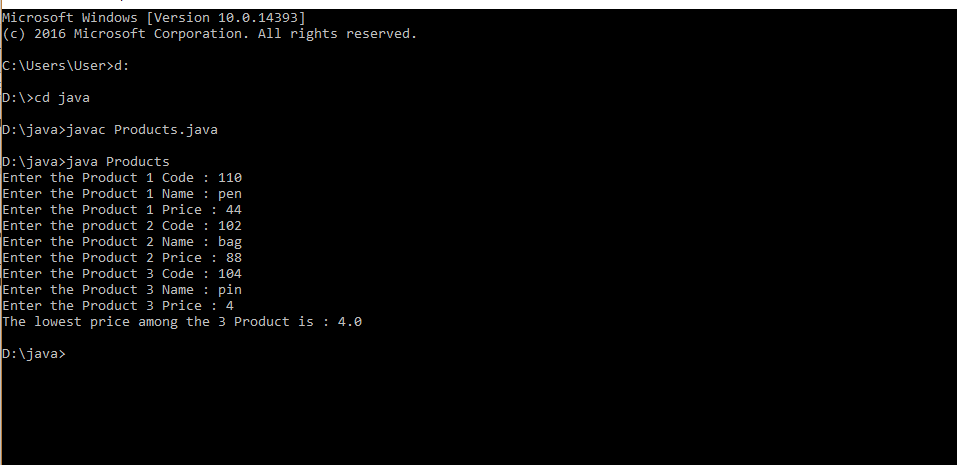
System.out.print("Enter the Product 3 Price : ");

p3.price = s.nextInt();

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

}

}

**Expected Output** 

**Experiment 5: CPU and its details - Using Inner classes**

**Aim**: Create CPU with attribute price. Create inner class Processor (no. of cores, manufacturer) and static nested class RAM (memory, manufacturer). Create an object of CPU and print information of Processor and RAM. (---No need to do now---).

**Algoritham**

1. start
2. declare the class CPU
3. initialize data price in the class CPU.
4. declare the inner class Processor with data member no. of cores, manufacturer
5. declare a member function getCache in class Processor.
6. declare the static nested class RAM with data member memory, manufacturer
7. declare a member function getClockSpeed.
8. Create an object of CPU : CPU cpu = new CPU();
9. CPU.Processor processor = cpu.new Processor();

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

1. print information of Processor and RAM.
2. stop

**Program:**

**Main.java**

class CPU

{

double price;

// nested class

class Processor{

double cores;

String manufacturer;

double getCache()

{

return 4.3;

}

}

// nested protected class

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();

// object of inner class Processor

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

// object of inner class RAM

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

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

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

}

}

**Expected Output** 