

OOPS LAB CYCLE-1

EXPERIMENT 1: SIMPLE JAVA PROGRAM

AIM: Java program to accept the marks of a student into 1D array and find total marks and percentage.

ALGORITHM:

1. Start
2. Create a public class Student_Subject
3. Use a one-dimensional array to store the marks of a student.
4. Read the no.of subjects(n) and initialize the array marks[] with 'n'.
5. Store the marks in the array marks[] using a for loop.
6. Find total mark and percentage.
7. Print the total and percentage.
8. Stop.

PROGRAM:

```
import java.util.Scanner;

class Student_Subject
{

    public static void main(String
    args[])

    {
        float
        sub1,sub2,sub3,sub4,sub5;

        double total, percentage;

        Scanner op=new
        Scanner(System.in);
```

```
System.out.println("Enter marks of five subjects");

System.out.print("Enter marks of subject1:");
sub1=op.nextFloat();

System.out.print("Enter marks of subject2:");

sub2=op.nextFloat();


System.out.print("Enter marks of subject3:");

sub3=op.nextFloat();


System.out.print("Enter marks of subject4:");

sub4=op.nextFloat();


System.out.print("Enter marks of subject5:");

sub5=op.nextFloat();


total = sub1+sub2+sub3+sub4+sub5;

percentage = (total / 500) * 100;

System.out.println("Total marks =" +total);

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

}

}
```

```

F:\software engineering>javac Student_Subject.java

F:\software engineering>java Student_Subject
Enter marks of five subjects
Enter marks of subject1:34
Enter marks of subject2:5
Enter marks of subject3:96
Enter marks of subject4:43
Enter marks of subject5:90
Total marks =268.0
Percentage = 53.6

F:\software engineering>

```

EXPERIMENT 2: COUNT NUMBER OF OCCURRENCES

AIM: Java program to count the number of occurrences of an element in an array

ALGORITHM:

1. Create a public class Count_Occurrence.
2. Define class with members n(int), item(int), count(int).
3. Get the data from user including number of elements and those elements.
4. Store each element in an 1D array i.e., a[i].
5. For each element in a[i] if a[i]==item then increment count, i.e., count++.
6. Print the count.
7. Stop.

PROGRAM:

```

import java.util.Scanner;

public class
Count_Occurrence

{   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);

}
}

```

```
F:\software engineering\java\count_occurrence.java
F:\software engineering>java Count_Occurrence
Enter no. of elements you want in array:5
Enter all the elements:
2
3
2
2
56
Enter the element of which you want to count number of occurrences:2
Number of Occurrence of the Element:3
F:\software engineering>
```

EXPERIMENT 3: MATRIX ADDITION

AIM: Read two matrices from the console and perform matrix addition.

ALGORITHM:

1. Start
2. Declare two matrices with 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 into $b[][]$.
5. Print two matrices.
6. Add the two matrices using for loop for $i=0$ to $i<\text{row}$ for $j=0$ to $j<\text{col}$
7. $a[i][j] + b[i][j]$ and store it in to the matrix at $c[i][j]$.
 $c[i][j] = a[i][j] + b[i][j];$
8. stop.

PROGRAM:

```
import java.util.Scanner;

public class Add_TwoMatrix

{

    public static void main(String[] args)

    {

        int p, q, m, n;

        Scanner s = new Scanner(System.in);

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

        p =

s.nextInt();    q

= s.nextInt();

        System.out.print(

"Enter number of

rows and

columns in

second matrix:");
```

```

        m = s.nextInt();

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 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 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

    {

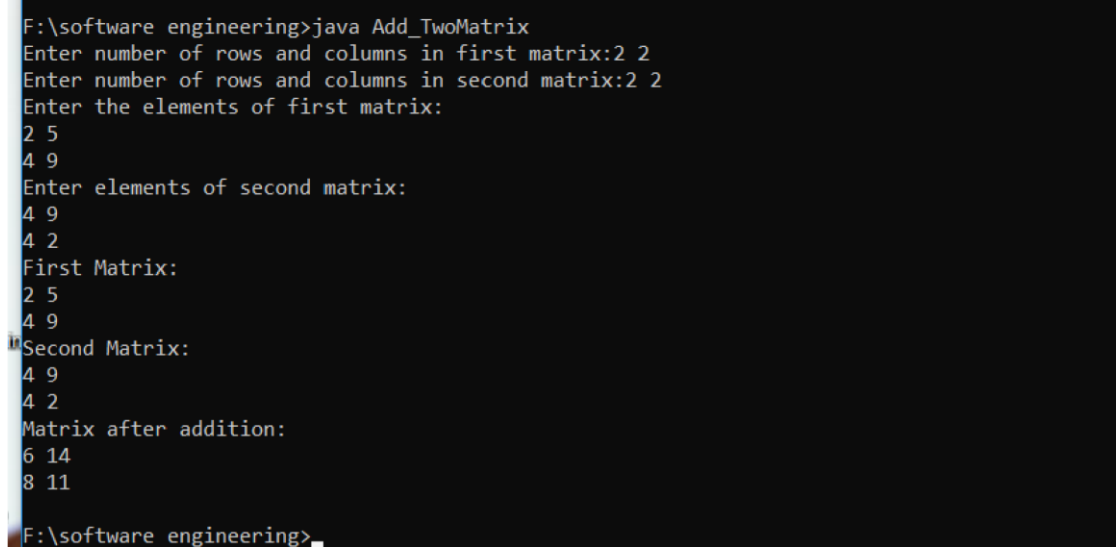
        System.out.println("Addition would not be possible");

    }

}

}

```



```

F:\software engineering>java Add_TwoMatrix
Enter number of rows and columns in first matrix:2 2
Enter number of rows and columns in second matrix:2 2
Enter the elements of first matrix:
2 5
4 9
Enter elements of second matrix:
4 9
4 2
First Matrix:
2 5
4 9
Second Matrix:
4 9
4 2
Matrix after addition:
6 14
8 11
F:\software engineering>

```

EXPERIMENT 4: COMPLEX NUMBER ADDITION

AIM: Add complex numbers.

ALGORITHM:

1. Start
2. Define class GFG
3. Define the real and imaginary part of a complex number
4. Define a method to initialize real and img.
5. Define a method to perform addition.
6. Define the main() method and create object of the class.
7. Call the methods and print the results.
8. Stop.

PROGRAM:

```
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);
}
}

```

OUTPUT:

```

F:\java>javac GFG.java

F:\java>java GFG
Complex number 1 : 3 + i2
Complex number 2 : 9 + i5
Sum of complex number : 12 + i7

F:\java>

```

EXPERIMENT 5: MATRIX IS SYMMETRIC OR NOT

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

ALGORITHM:

1. Start
2. Read no of rows and columns into integer variables m and n respectively.

3. Declare and initialize 2D array `a[m][n]`.
4. Store the matrix elements into the two-dimensional array `a[i][j]` using two for loops. `i` indicates row number, `j` indicates column index.
5. Display the input matrix.
6. if (`m != n`) then the given matrix is not a square matrix, so it can't be symmetric.
7. Otherwise, check if the matrix not equal to transpose matrix i.e., if (`a[i][j] != a[j][i]`). Print matrix is not symmetric.
8. Otherwise print matrix is symmetric.
9. Stop

PROGRAM:

```
import java.util.*; public class Symm

{

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++) { for (j
= 0; j < col; j++) { mat[i][j] = sc.nextInt();

}

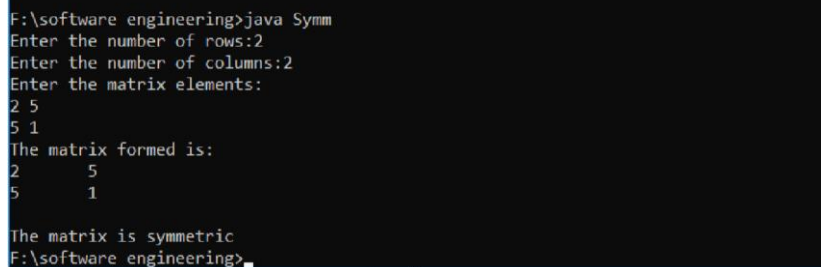
}

checkSymmetric(mat, row, col);

}

}

```



```

F:\software engineering>java Symm
Enter the number of rows:2
Enter the number of columns:2
Enter the matrix elements:
2 5
5 1
The matrix formed is:
2      5
5      1
The matrix is symmetric
F:\software engineering>

```

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 lowest price.

ALGORITHM:

1. Start
2. Define class Product_6 with members pcode (int), pname (String) and price(int).
3. Define a constructor to initialise pcode,pname and price.
4. Declare a member function void calculate and compare the lowest price of product.
5. Ask the user to enter pcode and pname,price of 3 products, and stored using object of class Product_6
6. Do appropriate calculation and display the value
7. Stop

PROGRAM:

/*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.*/

```
import java.util.*; public
class Product_6
{
    int pcode;
    String pname;
    int price;

    Product_6()
    {
        pcode=0;
        pname=null; price=0;
    }
    public static void calculate(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);
    Product p1= new Product();
    Product p2= new Product();
    Product p3= new Product();

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

```

OUTPUT:

```

Enter the Product 1 Code : 101
Enter the Product 1 Name : pen
Enter the Product 1 Price : 15
Enter the product 2 Code : 102
Enter the Product 2 Name : pencil
Enter the Product 2 Price : 10
Enter the Product 3 Code : 103
Enter the Product 3 Name : scale
Enter the Product 3 Price : 18
The lowest price among the 3 Product is : 10.0

```

EXPERIMENT 7: CPU AND ITS DETAILS–USING INNER CLASS

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.

ALGORITHM:

1. Define class CPU with member price of double datatype.
2. Define an inner class Processor with members cores (double) and manufacturer (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 Main
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.
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 memory type = ram.getClockSpeed ();
  10. End class Main

### **PROGRAM:**

/\*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.\*/

```
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());
 }
}
```

```
}
}
```

use -help for a list of possible options

```
F:\software engineering>javac Main.java
```

```
F:\software engineering>java Main
```

```
Processor Cache = 4.3
```

```
Ram Clock speed = 5.5
```

```
F:\software engineering>
```