


SOURCE CODE

```
public class WelcomeJava {  
    public static void main(String[] args) {  
        System.out.println("Welcome Java");  
    }  
}
```

OUTPUT



```
megha@MCA:~$ javac WelcomeJava.java  
megha@MCA:~$ java WelcomeJava  
Welcome Java  
megha@MCA:~$
```

Lab cycle:1
Experiment no:1

Date:13/03/2023

WELCOME JAVA

AIM: Write a java program to display the message “WELCOME JAVA”

ALGORITHM:

Step 1: Start

Step 2: Create a class called Welcome

Step 3:Display the message “WELCOME JAVA”

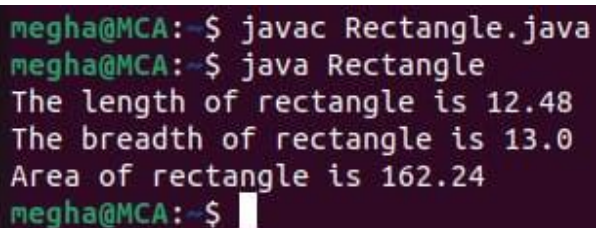
Step 3 : End

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
class Rectangle {  
    double length;  
    int breadth;  
    void setData() {  
        length=12.48;  
        System.out.println("The length of rectangle : "+length);  
        breadth=13;  
        System.out.println("The breadth of rectangle : "+breadth); }  
    void getArea() {  
        double area;  
        area=length*breadth;  
        System.out.println("The area of rectangle : "+area); }  
    public static void main(String args[]) {  
        Rectangle r1=new Rectangle();  
        r1.setData();  
        r1.getArea(); } }
```

OUTPUT



```
megha@MCA:~$ javac Rectangle.java  
megha@MCA:~$ java Rectangle  
The length of rectangle is 12.48  
The breadth of rectangle is 13.0  
Area of rectangle is 162.24  
megha@MCA:~$
```

Lab cycle:1
Experiment no:2

Date:13/03/2023

AREA OF RECTANGLE

AIM: Create a class Rectangle with instance variables length and breadth. Define a method setData() for setting values of instance variables and a method getArea() to return the area of rectangle using class. Find the area of the rectangle using the values length=12.48 and breadth=13

ALGORITHM:

Step 1: Start

Step 2: Create a class Rectangle with two instance variables length and breadth.

Step 3: Define a method setData() for setting values of instance variables and print their values.

Step 4: Define a method getArea() to find the area of rectangle.

Step 5: Get the area with length=12.48 and breadth=13, by calling the object of the class.

Step 6: Print the area

Step 7: End

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
import java.util.*;

class evenodd{

    public static void main(String args[]) {

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

        Scanner n=new Scanner(System.in);

        int num=n.nextInt();

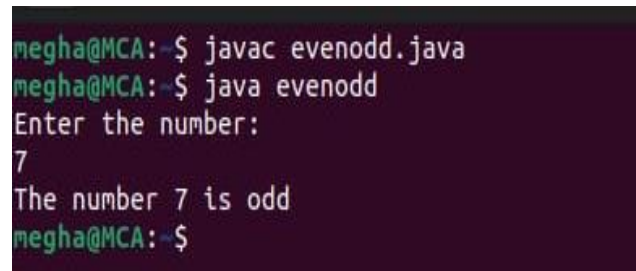
        if(num%2==0)    {

            System.out.println("The number "+num+" is even");    }

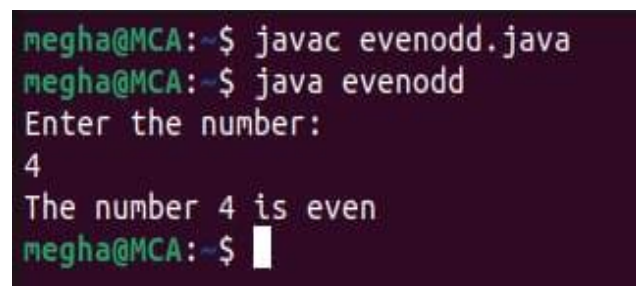
        else    {

            System.out.println("The number "+num+" is odd");    } } }
```

OUTPUT



```
megha@MCA:~$ javac evenodd.java
megha@MCA:~$ java evenodd
Enter the number:
7
The number 7 is odd
megha@MCA:~$
```



```
megha@MCA:~$ javac evenodd.java
megha@MCA:~$ java evenodd
Enter the number:
4
The number 4 is even
megha@MCA:~$
```

Lab cycle:1
Experiment no:3

Date:20/03/2023

EVEN OR ODD

AIM: Write a program to read integer from keyboard and check whether the number is even or odd

ALGORITHM:

Step 1: Start

Step 2: Declare a class Integer

Step 3:Read the input from user

Step 4: Check whether the given value is even or odd using `if(a%2==0)`

4.1: if the condition is true then print "The number is even."

4.2: otherwise print "The number is odd"

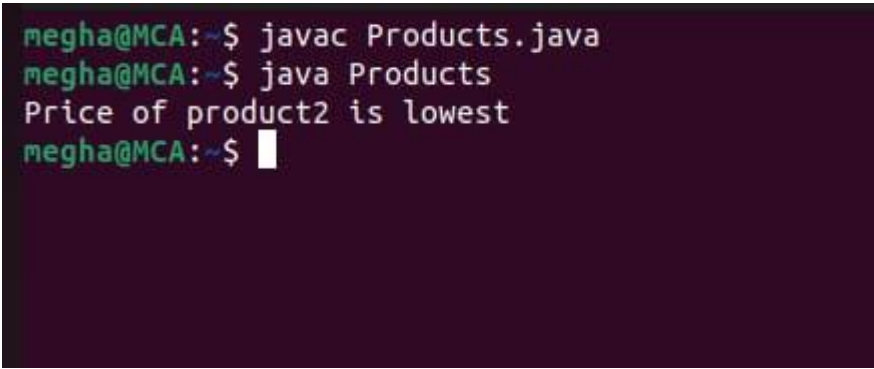
Step 5: End

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
public class Products{  
    int pcde;  
    String pnme;  
    double prce;  
    Products(int pcode,String pname,double price) {  
        pcde=pcode;  
        pnme=pname;  
        prce=price; }  
    public static void main(String args[]) {  
        p1= new Products(101,"product1",55.2);  
        Products p2= new Products(102,"product1",10.3);  
        Products p3= new Products(103,"product1",25.9);  
        if(p1.prce <p2.prce && p1.prce < p3.prce) {  
            System.out.println("Price of product1 is lowest"); }  
        else if(p2.prce<p1.prce && p2.prce < p3.prce) {  
            System.out.println("Price of product2 is lowest"); }  
        else if(p3.prce>p2.prce && p3.prce<p1.prce) {  
            System.out.println("Price of product3 is lowest") ; }}}
```

OUTPUT



```
megha@MCA:~$ javac Products.java  
megha@MCA:~$ java Products  
Price of product2 is lowest  
megha@MCA:~$
```

Lab cycle:1
Experiment no:4

Date:20/03/2023

PRODUCT USING CONSTRUCTOR

AIM: Define a class product with data members pcode ,pname, and price. Create three objects of the class and find the product having the lowest price.

ALGORITHM:

Step 1: Start

Step 2: Create a class product.

Step 3: Declare variables pname,pcd,prce.

Step 4: Create a parameterized constructor of the class with parameter pcode,pname,price.

Step 5: Assign values pcd=pcode, pnme=pname, ,prce= price.

Step 6: Create three objects for the class contain values, that is p1,p2,p3.

Step 7: Check the condition if price of 1st is less than price of 2nd and price of 1st less than price of 3rd .then,

7.1: print p1 is lowest price

7.2: check otherwise p2 price is less than p1 and p3.then print p2 is lowest.

7.3: check price of p3 greater than p2 and less than p1. Then print p3 is lowest.

Step 8: End

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```

import java.util.*;

class matrix{

    public static void main(String[] args)    {

        int r,c,i,j;

        Scanner s = new Scanner(System.in);

        System.out.println("Enter the row and column");

        r=s.nextInt();

        c=s.nextInt();

        int m1[][]=new int[r][c];

        int m2[][]=new int[r][c];

        int sum[][]=new int[r][c];

        System.out.println("Enter the 1st matrix");

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

            for(j=0;j<c;j++)

                m1[i][j]=s.nextInt();

        System.out.println("Enter the 2nd matrix");

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

            for(j=0;j<c;j++)

                m2[i][j]=s.nextInt();

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

            for(j=0;j<c;j++)

                sum[i][j]=m1[i][j]+m2[i][j];

        System.out.println("Sum of matrix:");

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

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

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

            System.out.println();    }    }    }

```

Lab cycle:1
Experiment no:5

Date:20/03/2023

MATRIX ADDITION

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

ALGORITHM:

Step 1: Start

Step 2: Create a class Matrix.

Step 3: Read the no . of row and columns.

Step 4: Declare three 2D arrays.

Step 5: Display the value of 1st matrix using for loop.

Step 6: Display the value of 2nd matrix using for loop.

Step 7: $\text{sum}[i][j] = m1[i][j] + m2[i][j]$

7.1: sum the values of first and second array.

7.2: placed in into 3rd array.

Step 9: End

OUTPUT

```
megha@MCA:~$ javac matrix.java
megha@MCA:~$ java matrix
Enter the row and column
2 2
Enter the 1st matrix
2 4
5 6
Enter the 2nd matrix
2 1
7 3
Sum of matrix:
4      5

12     9

megha@MCA:~$
```

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
import java.util.Scanner;

public class complexNoAddition{

    public static void main(String args[]){

        ComplexNo c1 = new ComplexNo();

        ComplexNo c2 = new ComplexNo();

        c1.addComplexNumbers(c2);    }    }

class ComplexNo{

    int r,i;

    Scanner read = new Scanner(System.in);

    ComplexNo() {

        System.out.println("Enter the complex no :[Enter real part first, then imaginary
part]");

        r = read.nextInt();

        i = read.nextInt();    }

    void addComplexNumbers(ComplexNo c2){

        int rr,ri;

        rr = r + c2.r;

        ri = i + c2.i;

        System.out.println("Sum of complex numbers = " + rr +"i"+ri);    }    }
```

OUTPUT

```
megha@MCA:~$ javac ComplexNoAddition.java
megha@MCA:~$ java ComplexNoAddition
Enter the complex no:[Enter real part first ,then imaginary part]
5 7
Enter the complex no:[Enter real part first ,then imaginary part]
9 1
Sum of complex numbers= 14+i8
megha@MCA:~$
```

Lab cycle:1
Experiment no:6

Date:24/03/2023

COMPLEX NUMBER ADDITION

AIM: Write a program to perform complex number addition.

ALGORITHM:

Step 1: Create a class ComplexNoAddition with a main method.

Step 2: Inside the main method, create two instances of ComplexNo class named c1 and c2.

Step 3: Prompt the user to enter the real and imaginary parts of the first complex number.

Step 4: Prompt the user to enter the real and imaginary parts of the second complex number.

Step 5: Calculate the sum of the complex numbers by adding the corresponding real and imaginary parts together.

Step 6: Print the sum of the complex numbers

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
import java.util.Scanner;

public class SymmetricMatrixExample {

    public static boolean isSymmetric(int[][] matrix) {

        int rows = matrix.length;

        int columns = matrix[0].length;

        if (rows != columns) {

            return false;    }

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

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

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

                    return false;    }    }    }

            return true;    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int rows = sc.nextInt();

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

        int columns = sc.nextInt();

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

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

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

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

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

        if(isSymmetric(matrix)) {

            System.out.println("Matrix is symmetric:");    }

        else {

            System.out.println("Matrix is not symmetric:");    }    }    }
```

Lab cycle:1
Experiment no:7

Date:24/03/2023

SYMMETRIC OR NOT

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

ALGORITHM:

Step 1: Initialize a boolean variable isSymmetric to true.

Step 2: Prompt the user to enter the number of rows and columns of the matrix.

Step 3: Create a 2D array to store the matrix elements.

Step 4: Prompt the user to enter the matrix elements row by row.

Step 5: Check if the matrix is square. If not, set isSymmetric to false.

Step 6: Iterate over the matrix elements and check if $\text{matrix}[i][j]$ is equal to $\text{matrix}[j][i]$ for all i and j . If not, set isSymmetric to false.

Step 7: If isSymmetric is true, print "Matrix is symmetric". Otherwise, print "Matrix is not symmetric".

OUTPUT

```
megha@MCA:~$ javac SymmetricMatrixExample.java
megha@MCA:~$ java SymmetricMatrixExample
Enter number of rows:3
Enter number of columns:3
Enter matrix elements:
2 3 6
3 4 5
6 5 9
Matrix is symmetric:
```

```
megha@MCA:~$ javac SymmetricMatrixExample.java
megha@MCA:~$ java SymmetricMatrixExample
Enter number of rows:3
Enter number of columns:3
Enter matrix elements:
3 4 9
2 5 1
6 7 3
Matrix is not symmetric:
megha@MCA:~$
```

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
import java.util.Scanner;

public class leap {

    public static void main(String[] args) {

        int startYear, endYear, i;

        Scanner in = new Scanner(System.in);

        System.out.print("Enter the Start Year:");

        startYear = in.nextInt();

        System.out.print("Enter the End Year:");

        endYear = in.nextInt();

        System.out.println("Leap years:");

        for (i = startYear; i <= endYear; i++)

        {

            if ( (0 == i % 4) && (0 != i % 100) || (0 == i % 400) ){

                System.out.println(i);

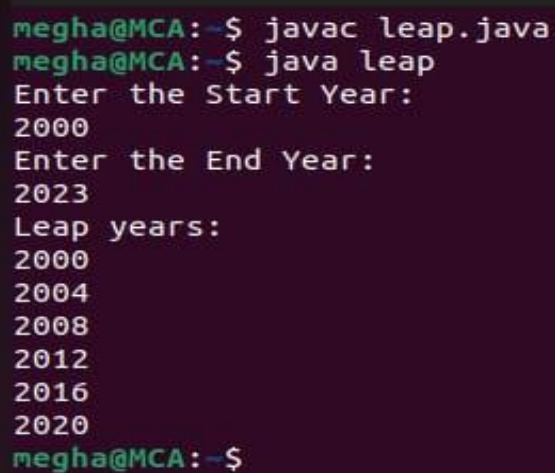
            }

        }

    }

}
```

OUTPUT



```
megha@MCA:~$ javac leap.java
megha@MCA:~$ java leap
Enter the Start Year:
2000
Enter the End Year:
2023
Leap years:
2000
2004
2008
2012
2016
2020
megha@MCA:~$
```

Lab cycle:1
Experiment no:8

Date:24/03/2023

LEAP YEAR

AIM: Write a program to print the leap years within the given range.

ALGORITHM:

Step 1: Declare a class leap year and declare three counter variables startYear, endYear and i

Step 2: Create a scanner object to get the input

Step 3: Get the start year and end year from user

Step 4: Print the leap years by using loop through between start year and end year.

Step 5: Use Boolean condition $i \% 4 == 0 \ \&\& \ i \% 100 != 0 \ || \ i \% 400 == 0$.

6.1: if the above condition is true then print each leap years.

Step 7: End

RESULT: The program has been executed successfully and output obtained.

SOURCE CODE

```
class cpu {
    double price;
    String name;
    cpu(double p,String name1) {
        price=p;
        name=name1; }
    class processor {
        int cores;
        String manufact;
        processor(int core,String m) {
            cores=core;
            manufact=m; } }
    static class ram {
        String memory;
        String manufact;
        ram(String mem,String man) {
            memory=mem;
            manufact=man; } }
    public static void main(String args[]) {
        cpu c=new cpu(1800,"intel");
        cpu.processor p=c.new processor(8,"intel");
        cpu.ram r=new cpu.ram("18gb","crucial");
        System.out.println("processor name="+c.name+"\nprocessor price="+c.price);
        System.out.println("processor cores="+p.cores+"\nprocessor manufact name="+p.manufact);
        System.out.println("memory capacity="+r.memory+"\nram manufact="+r.manufact);}}
```

Lab cycle:1
Experiment no:9

Date:24/03/2023

CPU

AIM: Create a CPU with attribute price, create an inner class processor(no .of cores ,manufactures) and static nested class RAM(memory, manufactures). Create an object of CPU and print information of processor and RAM.

ALGORITHM:

Step 1: Create an instance of the CPU class with a price of 1800 and the name "intel".

Step 2: Create an instance of the processor class with 8 cores and the manufacturer "intel".

Step 3: Create an instance of the ram class with a memory capacity of 18GB and the manufacturer "crucial".

Step 4: Print the processor name and price from the CPU instance.

Step 5: Print the processor cores and manufacturer from the processor instance.

Step 6: Print the memory capacity and manufacturer from the ram instance.

Step 7: End the program.

OUTPUT

```
megha@MCA:~$ javac cpu.java
megha@MCA:~$ java cpu
processor name=intel
processor price=1800.0
processor cores=8
processor manufact name=intel
memory capacity=18gb
ram manufact=crucial
megha@MCA:~$ █
```

RESULT: The program has been executed successfully and output obtained.