**1)READ MATRIX ELEMENTS FROM USER AND DISPLAY THE MATRIX**

**package** com.MatrixPrograms;

**import** java.util.Scanner;

//READ MATRIX ELEMENTS FROM USER AND DISPLAY THE MATRIX

**public** **class** ReadMatrix {

//READ MATRIX ELEMENTS FROM USER

**static** **int**[][]readMatrix(){

Scanner sc=**new** Scanner(System.***in***);

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

**int** row=sc.nextInt();

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

**int** col=sc.nextInt();

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

System.***out***.println("Enter "+row\*col+" elements");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

}

**return** mat;

}

//DISPLAYS MATRIX ELEMENTS

**static** **void** displayMatrix(**int** mat[][]) {

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

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

ReadMatrix rd=**new** ReadMatrix();

**int** [][]read=rd.*readMatrix*();

System.***out***.println("Matris is");

rd.*displayMatrix*(read);

}

}

**2)PRINT THE MATRIX ELEMENTS**

**package** com.MatrixPrograms;

//PRINT THE MATRIX ELEMENTS

**public** **class** PrintMatrix {

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

**int** mat[][]= {{10,20,30},{40,50,60},{70,80,90}};

System.***out***.println(mat.length);

System.***out***.println(mat[0].length);

System.***out***.println(mat[1][0]);

System.***out***.println(mat[2][2]);

System.***out***.println("Given matrix");

**for**(**int** i=0;i<mat.length;i++) {//FOR NO OF ROWS

**for**(**int** j=0;j<mat[i].length;j++) {//FOR NO OF ELEMENTS EACH ROW/NO OF COLUMNS

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

}

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

}

}

}

**3)SUM OF ALL MATRIX ELEMENTS**

**package** com.MatrixPrograms;

//SUM OF ALL MATRIX ELEMENTS

**public** **class** MatrixElementsSum {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** sum=*printSum*(mat);

System.***out***.println("Sum of matrix elements is "+sum);

}

**private** **static** **int** printSum(**int**[][] mat) {

**int** sum=0;

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

sum=sum+mat[i][j];

}

}

**return** sum;

}

}

**4)SUM OF EVEN AND ODD NUMBER IN MATRIX**

**package** com.MatrixPrograms;

//SUM OF EVEN AND ODD NUMBER IN MATRIX

**public** **class** MatrixEvenOddSum {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** sum[]=*evenoddSum*(mat);

System.***out***.println("Even number sum "+sum[0]);

System.***out***.println("Odd number sum is "+sum[1]);

}

**private** **static** **int**[] evenoddSum(**int**[][] mat) {

**int** es=0,os=0;

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

**if**(mat[i][j]%2==0)

es=es+mat[i][j];

**else**

os=os+mat[i][j];

}

}

**int** s[]= {es,os};

**return** s;

}

}

**5)COUNT EVEN AND ODD NUMBER IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//COUNT EVEN AND ODD NUMBER IN A GIVEN MATRIX

**public** **class** MatrixEvenOddCount {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** count[]=*evenoddCount*(mat);

System.***out***.println("Even elemets count is "+count[0]);

System.***out***.println("Odd elements count is "+count[1]);

}

**private** **static** **int**[] evenoddCount(**int**[][] mat) {

**int** ec=0,oc=0;

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

**if**(mat[i][j]%2==0)

ec++;

**else**

oc++;

}

}

**int** c[]= {ec,oc};

**return** c;

}

}

**6)COUNT HOW MANY PRIME NUMBERS PRESENT IN A MATRIX**

**package** com.MatrixPrograms;

//COUNT HOW MANY PRIME NUMBERS PRESENT IN A MATRIX

**public** **class** MatrixPrimeElementsCount{

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** pc=*primeCount*(mat);

System.***out***.println("Total prime numbers count "+pc);

}

**private** **static** **int** primeCount(**int**[][] mat) {

**int** count=0;

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

**if**(*isPrime*(mat[i][j]))

count++;

}

}

**return** count;

}

**static** **boolean** isPrime(**int** n) {

**if**(n<2)

**return** **false**;

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

**if**(n%i==0)

**return** **false**;

}

**return** **true**;

}

}

**7)ADD TWO MATRIX**

**package** com.MatrixPrograms;

//ADD TWO MATRIX

**public** **class** AddTwoMatrix {

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

**int** mat1[][]= {{11,12,13},{4,5,6},{7,8,9}};

**int** mat2[][]= {{3,2,1},{6,5,4},{9,8,7}};

**int** sum[][]=*addMatrix*(mat1,mat2);

**if**(sum==**null**)

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

**else** {

System.***out***.println("Added matrix is ");

**for**(**int** i=0;i<sum.length;i++) {

**for**(**int** j=0;j<sum[i].length;j++) {

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

}

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

}

}

}

**private** **static** **int**[][] addMatrix(**int**[][] mat1, **int**[][] mat2) {

**if**(mat1.length!=mat2.length||mat1[0].length!=mat2[0].length)

**return** **null**;

**int** mat3[][]=**new** **int**[mat1.length][mat1[0].length];

**for**(**int** i=0;i<mat3.length;i++) {

**for**(**int** j=0;j<mat3[i].length;j++) {

mat3[i][j]=mat1[i][j]+mat2[i][j];

}

}

**return** mat3;

}

}

**8)SEARCH MATRIX ELEMENT**

**package** com.MatrixPrograms;

//SEARCH MATRIX ELEMENT

**public** **class** SearchMatrixElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** n=mat.length;

**boolean** rs=*searchElement*(mat,n,-1);

**if**(rs)

System.***out***.println("Element is found");

**else**

System.***out***.println("Element is not found");

}

**private** **static** **boolean** searchElement(**int**[][] mat,**int** n,**int** ele) {

**if**(n==0)

**return** **false**;

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

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

**if**(mat[i][j]==ele)

**return** **true**;

}

}

**return** **false**;

}

}

**9)SEARCH MATRIX ELEMENT AND RETURN SPECIFIED INDEX POSITION OF THIS ELEMENT**

**package** com.MatrixPrograms;

//SEARCH MATRIX ELEMENT AND RETURN SPECIFIED INDEX POSITION OF THIS ELEMENT

**public** **class** MatrixElementSearch {

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

**int** mat[][]= {{1,2,3,10},{4,5,6,12},{7,8,9,13}};

**int** m=mat.length;

**int** n=mat[0].length;

*serachElement*(mat,m,n,12);

}

**private** **static** **int** serachElement(**int**[][] mat,**int** m,**int** n,**int** ele) {

**if**(m==0||n==0)

**return** -1;

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

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

**if**(mat[i][j]==ele) {

System.***out***.println("Element is found at("+i+","+j+")\n");

**return** 1;

}

}

}

System.***out***.println("Element is not found");

**return** 0;

}

}

**10)M X N MATRIX TRANSPOSE**

**package** com.MatrixPrograms;

//M X N MATRIX TRANSPOSE

**public** **class** TransposeMatrix {

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

**int** mat[][]= {{1,2,3,4},{5,6,7,8},{9,10,11,12}};

**int** rs[][]=*transposeMatrix*(mat);

System.***out***.println("Transpose Matrix");

**for**(**int** i=0;i<rs.length;i++) {

**for**(**int** j=0;j<rs[i].length;j++) {

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

}

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

}

}

**private** **static** **int**[][] transposeMatrix(**int**[][] mat) {

**int**[][]res=**new** **int**[mat[0].length][mat.length];

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

res[j][i]=mat[i][j];

}

}

**return** res;

}

}

**11)N X N MATRIX TRANSPOSE**

**package** com.MatrixPrograms;

//N X N MATRIX TRANSPOSE

**public** **class** MatrixTranspose {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

*transposeMatrix*(mat);

System.***out***.println("Transpose Matrix");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

**private** **static** **void** transposeMatrix(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=i+1;j<mat[i].length;j++) {

**int** temp=mat[i][j];

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

mat[j][i]=temp;

}

}

}

}

**12)PRINT ROWWISE BIGGEST ELEMENT IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//PRINT ROWWISE BIGGEST ELEMENT IN A GIVEN MATRIX

**public** **class** RowWiseBiggestElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** rb[]=*rowBiggest*(mat);

System.***out***.println("Rowwise Biggest Element in each row");

**for**(**int** i=0;i<rb.length;i++) {

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

}

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

}

**private** **static** **int**[] rowBiggest(**int**[][] mat) {

**int** big[]=**new** **int**[mat.length];

**for**(**int** i=0;i<mat.length;i++) {

**int** max=mat[i][0];

**for**(**int** j=0;j<mat[i].length;j++) {

**if**(mat[i][j]>max)

max=mat[i][j];

}

big[i]=max;

}

**return** big;

}

}

**13)PRINT ROWWISE SMALLEST ELEMENT IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//PRINT ROWWISE SMALLEST ELEMENT IN A GIVEN MATRIX

**public** **class** RowWiseSmallestElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** rs[]=*rowSmallest*(mat);

System.***out***.println("Smallest element in each row");

**for**(**int** i=0;i<mat.length;i++) {

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

}

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

}

**private** **static** **int**[] rowSmallest(**int**[][] mat) {

**int** small[]=**new** **int**[mat.length];

**for**(**int** i=0;i<mat.length;i++) {

**int** min=mat[i][0];

**for**(**int** j=0;j<mat[i].length;j++) {

**if**(mat[i][j]<min)

min=mat[i][j];

}

small[i]=min;

}

**return** small;

}

}

**14)PRINT COLUMN WISE BIGGEST ELEMENT IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//PRINT COLUMN WISE BIGGEST ELEMENT IN A GIVEN MATRIX

**public** **class** ColumnWiseBiggestElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** cm[]=*columnBiggest*(mat);

System.***out***.println("Biggest element of each column");

**for**(**int** i=0;i<cm.length;i++) {

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

}

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

}

**private** **static** **int**[] columnBiggest(**int**[][] mat) {

**int** big[]=**new** **int**[mat[0].length];

**for**(**int** i=0;i<mat[0].length;i++) {

**int** max=mat[0][i];

**for**(**int** j=0;j<mat.length;j++) {

**if**(mat[j][i]>max)

max=mat[j][i];

}

big[i]=max;

}

**return** big;

}

}

**15)PRINT COLUMN WISE SMALLEST ELEMENT IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//PRINT COLUMN WISE SMALLEST ELEMENT IN A GIVEN MATRIX

**public** **class** ColumnWiseSmallestElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** se[]=*columnSmallest*(mat);

System.***out***.println("Smallest element in each row");

**for**(**int** i=0;i<se.length;i++) {

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

}

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

}

**private** **static** **int**[] columnSmallest(**int**[][] mat) {

**int** small[]= **new** **int**[mat[0].length];

**for**(**int** i=0;i<mat[0].length;i++) {

**int** min=mat[0][i];

**for**(**int** j=0;j<mat.length;j++) {

**if**(mat[j][i]<min)

min=mat[j][i];

}

small[i]=min;

}

**return** small;

}

}

**16)PRINT SUM OF EACH ROW ELEMENTS IN A MATRIX**

**package** com.MatrixPrograms;

//PRINT SUM OF EACH ROW ELEMENTS IN A MATRIX

**public** **class** RowwiseElementSum {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** s[]=*rowSum*(mat);

System.***out***.println("Rowwise elements sum");

**for**(**int** i=0;i<s.length;i++) {

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

}

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

}

**private** **static** **int**[] rowSum(**int**[][] mat) {

**int** sum[]=**new** **int**[mat.length];

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

sum[i]=sum[i]+mat[i][j];

}

}

**return** sum;

}

}

**17)PRINT SUM OF EACH COLUMN ELEMENTS IN A MATRIX**

**package** com.MatrixPrograms;

//PRINT SUM OF EACH COLUMN ELEMENTS IN A MATRIX

**public** **class** ColumnWiseElementSum {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** cs[]=*columnwiseSum*(mat);

System.***out***.println("Column wise element sum");

**for**(**int** i=0;i<cs.length;i++) {

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

}

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

}

**private** **static** **int**[] columnwiseSum(**int**[][] mat) {

**int** sum[]=**new** **int**[mat[0].length];

**for**(**int** i=0;i<mat[0].length;i++) {

**for**(**int** j=0;j<mat.length;j++) {

sum[i]=sum[i]+mat[j][i];

}

}

**return** sum;

}

}

**18)ROWWISE REVERSE THE MATRIX ELEMENTS**

**package** com.MatrixPrograms;

//ROWWISE REVERSE THE MATRIX ELEMENTS

**public** **class** RowwiseReverse {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

*rowwiseReverse*(mat);

System.***out***.println("Rowwise reverse matrix");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

**private** **static** **void** rowwiseReverse(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++) {

**int** f=0;

**int** l=mat[0].length-1;

**while**(f<l) {

**int** temp=mat[i][f];

mat[i][f]=mat[i][l];

mat[i][l]=temp;

f++;

l--;

}

}

}

}

**20)COLUMN WISE REVERSE THE MATRIX ELEMENTS**

**package** com.MatrixPrograms;

//COLUMN WISE REVERSE THE MATRIX ELEMENTS

**public** **class** ColumnWiseReverse {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

*rowwiseReverse*(mat);

System.***out***.println("Column wise reverse matrix");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

**private** **static** **void** columnwiseReverse(**int**[][] mat) {

**for**(**int** i=0;i<mat[0].length;i++) {

**int** f=0;

**int** l=mat.length-1;

**while**(f<l) {

**int** temp=mat[f][i];

mat[f][i]=mat[l][i];

mat[l][i]=temp;

f++;

l--;

}

}

}

}

**21)ROTATE THE MATRIX 90 DEGREE AT CLOCKWISE DIRECTION**

**package** com.MatrixPrograms;

//ROTATE THE MATRIX 90 DEGREE AT CLOCKWISE DIRECTION

**public** **class** Rotate90ClockWise {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

*rotateClockwise*(mat);

System.***out***.println("Matrix after 90 degree clockwise rotation ");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

**private** **static** **void** rotateClockwise(**int**[][] mat) {

*transposeMatrix*(mat);

*reverseRowwise*(mat);

}

**private** **static** **void** reverseRowwise(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++) {

**int** f=0;

**int** l=mat[0].length-1;

**while**(f<l) {

**int** temp=mat[i][f];

mat[i][f]=mat[i][l];

mat[i][l]=temp;

f++;

l--;

}

}

}

**private** **static** **void** transposeMatrix(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=i+1;j<mat[i].length;j++) {

**int** temp=mat[i][j];

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

mat[j][i]=temp;

}

}

}

}

**22)ROTATE THE MATRIX 90 DEGREE AT ANTI-CLOCKWISE DIRECTION**

**package** com.MatrixPrograms;

//ROTATE THE MATRIX 90 DEGREE AT ANTI-CLOCKWISE DIRECTION

**public** **class** Rotate90AntiClockWise {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

*rotateAntiClockwise*(mat);

System.***out***.println("Matrix after 90 degree Anti-clockwise rotation ");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

**private** **static** **void** rotateAntiClockwise(**int**[][] mat) {

*transposeMatrix*(mat);

*reverseColumnwise*(mat);

}

**private** **static** **void** reverseColumnwise(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++) {

**int** f=0;

**int** l=mat.length-1;

**while**(f<l) {

**int** temp=mat[f][i];

mat[f][i]=mat[l][i];

mat[l][i]=temp;

f++;

l--;

}

}

}

**private** **static** **void** transposeMatrix(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=i+1;j<mat[i].length;j++) {

**int** temp=mat[i][j];

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

mat[j][i]=temp;

}

}

}

}

**23)PRODUCT OF TWO MATRIX**

**package** com.MatrixPrograms;

//PRODUCT OF TWO MATRIX

**public** **class** MatrixProduct {

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

**int** mat1[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** mat2[][]= {{10,11,12},{13,14,15},{16,17,18}};

**int** p[][]=*matrixProduct*(mat1,mat2);

System.***out***.println("Ptoduct of two matrix");

**for**(**int** i=0;i<p.length;i++) {

**for**(**int** j=0;j<p[i].length;j++) {

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

}

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

}

}

**private** **static** **int**[][] matrixProduct(**int**[][] mat1, **int**[][] mat2) {

**if**(mat1[0].length!=mat2.length)

**return** **null**;

**int** mal[][]=**new** **int**[mat1.length][mat2[0].length];

**for**(**int** i=0;i<mat1.length;i++) {

**for**(**int** j=0;j<mat2[0].length;j++) {

**for**(**int** k=0;k<mat2.length;k++) {

mal[i][j]=mal[i][j]+mat1[i][k]\*mat2[k][j];

}

}

}

**return** mal;

}

}

**24)RETURN DIAGONALIZE ELEMENTS SUM OF A GIVEN MATRIX**

**package** com.MatrixPrograms;

//RETURN DIAGONALIZE ELEMENTS SUM OF A GIVEN MATRIX

**public** **class** DiagonalElementsSum {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** de[]=*digonalizeElementSum*(mat);

System.***out***.println("Diagonalize Elements sum");

System.***out***.println(de[0]);

System.***out***.println(de[1]);

}

**private** **static** **int**[] digonalizeElementSum(**int**[][] mat) {

**int** pSum=0,sSum=0;

**for**(**int** i=0;i<mat.length;i++) {

pSum=pSum+mat[i][i];

sSum=sSum+mat[i][mat.length-1-i];

}

**int** dSum[]= {pSum,sSum};

**return** dSum;

}

}

**25)RETURN DIAGONALIZE BIGGEST ELEMENT IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//RETURN DIAGONALIZE BIGGEST ELEMENT IN A GIVEN MATRIX

**public** **class** DiagonalizeBiggestElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** db[]=*diagonalizeBiggest*(mat);

System.***out***.println("Diagonalize biggest element");

System.***out***.println(db[0]);

System.***out***.println(db[1]);

}

**private** **static** **int**[] diagonalizeBiggest(**int**[][] mat) {

**int** pBig=mat[0][0];

**int** sBig=mat[0][mat.length-1];

**for**(**int** i=1;i<mat.length;i++) {

**if**(mat[i][i]>pBig)

pBig=mat[i][i];

**if**(mat[i][mat.length-1-i]>sBig)

sBig=mat[i][mat.length-1-i];

}

**int** dBig[]= {pBig,sBig};

**return** dBig;

}

}

**26)RETURN DIAGONALIZE SMALLEST ELEMENT IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//RETURN DIAGONALIZE SMALLEST ELEMENT IN A GIVEN MATRIX

**public** **class** DiagonalizeSmallestElement {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** ds[]=*diagonalizeSmallest*(mat);

System.***out***.println("Diagonalize Smallest element");

System.***out***.println(ds[0]);

System.***out***.println(ds[1]);

}

**private** **static** **int**[] diagonalizeSmallest(**int**[][] mat) {

**int** pSmall=mat[0][0];

**int** sSmall=mat[0][mat.length-1];

**for**(**int** i=1;i<mat.length;i++) {

**if**(mat[i][i]<pSmall)

pSmall=mat[i][i];

**if**(mat[i][mat.length-1-i]<sSmall)

sSmall=mat[i][mat.length-1-i];

}

**int** dSmall[]= {pSmall,sSmall};

**return** dSmall;

}

}

**27)REVERSE DIAGONALS ELEMENTS IN A GIVEN MATRIX**

**package** com.MatrixPrograms;

//REVERSE DIAGONALS ELEMENTS IN A GIVEN MATRIX

**public** **class** DiagonalizeElementsReverse {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

*diagonalizeElementsReverse*(mat);

System.***out***.println("Diagonalize elements reverse matrix");

**for**(**int** i=0;i<mat.length;i++) {

**for**(**int** j=0;j<mat[i].length;j++) {

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

}

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

}

}

**private** **static** **void** diagonalizeElementsReverse(**int**[][] mat) {

**int** f=0;

**int** l=mat.length-1;

**while**(f<l) {

**int** temp=mat[f][f];

mat[f][f]=mat[l][l];

mat[l][l]=temp;

temp=mat[f][l];

mat[f][l]=mat[l][f];

mat[l][f]=temp;

f++;

l--;

}

}

}

**28)PRINT A GIVEN MATRIX IN SPIRAL FORM(CLOCK WISE DIRECTION**)

**package** com.MatrixPrograms;

//PRINT A GIVEN MATRIX IN SPIRAL FORM(CLOCK WISE DIRECTION)

**public** **class** SpiralMatrix {

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

**int** mat[][]= {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};

*spiralMatrix*(mat);

}

**private** **static** **void** spiralMatrix(**int**[][] mat) {

**int** n=mat.length;

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

{

**for**(**int** k=i;k<j;k++)//FOW FAST ROW ELEMENTS

{

System.***out***.print(mat[i][k]+" ");

}

**for**(**int** k=i;k<j;k++)//FOR LAST COLUMN ELEMENTS

{

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

}

**for**(**int** k=j;k>i;k--)//FOR LAST ROW ELEMENTS

{

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

}

**for**(**int** k=j;k>i;k--)//FOR FAST COLUMN ELEMENTS

{

System.***out***.print(mat[k][i]+" ");

}

}

**if**(n%2==1)

System.***out***.println(mat[n/2][n/2]);

}

}

**29)PRINT A GIVEN MATRIX IN SPIRAL FORM(ANTI-CLOCK WISE DIRECTION)**

**package** com.MatrixPrograms;

//PRINT A GIVEN MATRIX IN SPIRAL FORM(ANTI-CLOCK WISE DIRECTION)

**public** **class** MatrixSpiral {

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

**int** mat[][]= {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};

*spiralMatrix*(mat);

}

**private** **static** **void** spiralMatrix(**int**[][] mat) {

**int** n=mat.length;

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

{

**for**(**int** k=i;k<j;k++)//FOW FAST COLUMN ELEMENTS

{

System.***out***.print(mat[k][i]+" ");

}

**for**(**int** k=i;k<j;k++)//FOR LAST ROW ELEMENTS

{

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

}

**for**(**int** k=j;k>i;k--)//FOR LAST COLUMN ELEMENTS

{

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

}

**for**(**int** k=j;k>i;k--)//FOR FAST ROW ELEMENTS

{

System.***out***.print(mat[i][k]+" ");

}

}

**if**(n%2==1)

System.***out***.println(mat[n/2][n/2]);

}

}

**30)SEARCH AN ELEMENT IN A ROW AND COLUMN WISE SORTED MATRIX**

**package** com.MatrixPrograms;

//SEARCH AN ELEMENT IN A ROW AND COLUMN WISE SORTED MATRIX

**public** **class** SearchRowColumnSortedMatrix

{

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

{

**int** mat[][]= {{1,4,7,11,15},{2,5,8,12,19},{3,6,9,16,22},{10,13,14,17,24}};

**boolean** se=*searchElement*(mat,8);

**if**(se)

System.***out***.println("Element is found");

**else**

System.***out***.println("Element is not found");

}

**private** **static** **boolean** searchElement(**int**[][] mat,**int** target)

{

**int** i=0;

**int** j=mat[0].length-1;

**while**(i<mat.length&&j>=0)

{

**if**(mat[i][j]==target)

**return** **true**;

**else** **if**(mat[i][j]>target)

j--;

**else**

i++;

}

**return** **false**;

}

}

**31)DISPLAY THE LOWER TRAINGULAR MATRIX**

**package** com.MatrixPrograms;

//DISPLAY THE LOWER TRAINGULAR MATRIX

//A square matrix is called lower triangular if all the entries above the main diagonal are zero.

**public** **class** DisplayLowerTraingularMatrix {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

System.***out***.println("Lower Traingular Matrix is");

*lowerTraingularMatrix*(mat);

}

**private** **static** **void** lowerTraingularMatrix(**int**[][] mat) {

**if**(mat.length!=mat[0].length)

{

System.***out***.println("Matrix should be square matrix");

}

**else**

{

**for**(**int** i=0;i<mat.length;i++)

{

**for**(**int** j=0;j<mat[i].length;j++)

{

**if**(j>i)

{

System.***out***.print("0"+" ");

}

**else**

{

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

}

}

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

}

}

}

}

**32)DISPLAY THE UPPER TRAINGULAR MATRIX**

**package** com.MatrixPrograms;

//DISPLAY THE UPPER TRAINGULAR MATRIX

//A square matrix is called upper triangular if all the entries below the main diagonal are zero.

**public** **class** DisplayUpperTraingularMatrix {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

System.***out***.println("Upper Traingular Matrix is");

*upperTraingularMatrix*(mat);

}

**private** **static** **void** upperTraingularMatrix(**int**[][] mat) {

**if**(mat.length!=mat[0].length)

System.***out***.println("Matrix should be square matrix");

**else**

{

**for**(**int** i=0;i<mat.length;i++)

{

**for**(**int** j=0;j<mat[i].length;j++)

{

**if**(i>j)

System.***out***.print("0"+" ");

**else**

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

}

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

}

}

}

}

**33)Java Program to determine whether a given matrix is an identity matrix**

**package** com.MatrixPrograms;

//Java Program to determine whether a given matrix is an identity matrix

**public** **class** IdentityMatrix {

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

**int** mat[][]= {{1,0,0},{0,1,0},{0,0,1}};

**boolean** rs=*isIdentity*(mat);

**if**(rs)

System.***out***.println("Identity matrix");

**else**

System.***out***.println("Not an Identity matrix");

}

**private** **static** **boolean** isIdentity(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++)

{

**for**(**int** j=0;j<mat[i].length;j++)

{

**if**(i==j&&mat[i][j]!=1)

**return** **false**;

**else** **if**(i!=j&&mat[i][j]!=0)

**return** **false**;

}

}

**return** **true**;

}

}

**34)Java Program to determine whether a given matrix is a sparse matrix**

**package** com.MatrixPrograms;

//Java Program to determine whether a given matrix is a sparse matrix

/\*A matrix is said to be sparse matrix if most of the elements of that matrix are 0.

It implies that it contains very less non-zero elements.

\*/

**public** **class** SparseMatrix {

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

**int** mat[][]= {{1,2,3},{0,0,0},{0,0,0}};

*isSparse*(mat);

}

**private** **static** **void** isSparse(**int**[][] mat) {

**int** count=0;

**int** size=mat.length\*mat[0].length;

**for**(**int** i=0;i<mat.length;i++)

{

**for**(**int** j=0;j<mat[i].length;j++)

{

**if**(mat[i][j]==0)

count++;

}

}

**if**(count>(size/2))

System.***out***.println("Sparse Matrix");

**else**

System.***out***.println("Not a Sparse Matrix");

}

}

**35)Java Program print Identity Matrix**

**package** com.MatrixPrograms;

//Java Program print Identity Matrix

**public** **class** PrintIdentityMatrix {

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

**int** size=4;

System.***out***.println("Identity Matrix is");

*printIdentity*(size);

}

**private** **static** **void** printIdentity(**int** size) {

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

{

**for**(**int** j=0;j<size;j++)

{

**if**(i==j)

System.***out***.print("1"+"");

**else**

System.***out***.print("0"+"");

}

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

}

}

}

**36)Java Program to Print matrix in snake pattern (OR) Print the Matrix elements in Zig-Zag form**

**package** com.MatrixPrograms;

//Java Program to Print matrix in snake pattern

**public** **class** PrintElementsSnakePattern {

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

**int** mat[][]= {{1,2,3,4},{5,6,7,8},{9,10,11,12}};

*snakePattern*(mat);

}

**private** **static** **void** snakePattern(**int**[][] mat) {

**for**(**int** i=0;i<mat.length;i++)

{

// If current row is even, print from

// left to right

**if**(i%2==0)

{

**for**(**int** j=0;j<mat[i].length;j++)

{

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

}

}

// If current row is odd, print from

// right to left

**else**

{

**for**(**int** j=mat[i].length-1;j>=0;j--)

{

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

}

}

}

}

}

**37)Print Elements in Sorted Order using Row-Column Wise Sorted Matrix in Java**

**package** com.MatrixPrograms;

**import** java.util.Arrays;

//Print Elements in Sorted Order using Row-Column Wise Sorted Matrix in Java

**public** **class** PrintMatrixElementSortedOrder {

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

**int** mat[][]= {{1,2,3},

{4,5,6},

{7,8,9}};

*sortedOrder*(mat);

}

**private** **static** **void** sortedOrder(**int**[][] mat) {

**int** m=mat.length;

**int** n=mat[0].length;

**int** result[]=**new** **int**[m\*n];

**int** x=0;

**for**(**int** i=0;i<mat.length;i++)

{

**for**(**int** j=0;j<mat[i].length;j++)

{

result[x++]=mat[i][j];

}

}

**int** size=m\*n;

Arrays.*sort*(result);

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

{

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

}

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

}

}

**38)Search an Element in a Matrix in Java**

**package** com.MatrixPrograms;

//Search an Element in a Matrix in Java

**public** **class** MatrixElementSearch {

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

**int** mat[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** ele=7;

*searchElement*(mat,ele);

}

**private** **static** **void** searchElement(**int**[][] mat, **int** ele) {

**int** flag=0;

**for**(**int** i=0;i<mat.length;i++)

{

**for**(**int** j=0;j<mat[0].length;j++)

{

**if**(mat[i][j]==ele)

{

System.***out***.println("Element is found("+i+","+j+")position");

flag=1;

**break**;

}

}

**if**(flag==1)

**break**;

}

**if**(flag==0)

System.***out***.println("Element is not found");

}

}

**39)Java Program to determine whether two matrices are equal**

**package** com.MatrixPrograms;

//Java Program to determine whether two matrices are equal

**public** **class** CheckMatrixEqualOrNot {

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

**int** a[][]= {{1,2,3},{4,5,6},{7,8,9}};

**int** b[][]= {{1,2,3,5},{4,5,6,3},{7,8,9,10}};

**boolean** rs=*isEqual*(a,b);

**if**(rs)

System.***out***.println("Matrix are equal matrix");

**else**

System.***out***.println("Matrix are not equal matrix");

}

**private** **static** **boolean** isEqual(**int**[][] a, **int**[][] b) {

**int** row1=a.length,col1=a[0].length;

**int** row2=b.length,col2=b[0].length;

**if**(row1!=row2||col1!=col2)

**return** **false**;

**for**(**int** i=0;i<row1;i++)

{

**for**(**int** j=0;j<col1;j++)

{

**if**(a[i][j]!=b[i][j])

{

**return** **false**;

}

}

}

**return** **true**;

}

}

**40)Java program to print all the Boundry elements of the given matrix**

**package** com;

//Java program to print all the Boundry elements of the given matrix

**public** **class** PrintAllBoundryElements {

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

**int** mat[][]= {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};

*printBoundaryElements*(mat);

}

**private** **static** **void** printBoundaryElements(**int**[][] mat) {

**int** i=0;

**int** j=0;

**for**( j=0;j<mat.length;j++)

{

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

}

j=mat.length-1;

**for**( i=1;i<mat.length;i++)

{

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

}

i=mat.length-1;

**for**(j=mat.length-2;j>=0;j--)

{

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

}

j=0;

**for**(i=mat.length-2;i>0;i--)

{

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

}

}

}