Matrices in Java Language

In this presentation, we explore the definition, application, and operations of matrices in Java. Join us on this journey to unlock the power of matrices!

A

by AMLAN PRASAD SAHOO

Introduction

Data is a valuable asset, and proper storage and access are crucial to avoid costly mistakes. Companies like Amazon, Spotify, and Netflix use data structures like arrays to efficiently manage their data. Arrays are static structures that can be single or multidimensional, similar to matrices. Since I am a computer engineering student, I used Java to perform operations on matrices and created a menu-based program.

What are Matrices?

Definition

Matrices are rectangular arrays of numbers arranged in rows and columns. They are used to represent linear equations and transformations.

Application in software industries

Matrices are useful when working with graphics, machine learning, data storage, and scientific simulations.

Creating a Matrix in Java

In Java, we can create matrices using arrays. Simply create an arbitrary sized dimensional array and fill it with values.

Operations on Matrices

1

Addition of two matrices

Matrices can be added or subtracted from other matrices of the same size by adding or subtracting the corresponding elements.

2

Multiplication of two matrices

Matrix multiplication is performed by multiplying the elements of one matrix with the corresponding elements of the other matrix and summing up the results.

3

Finding Determinants

The determinant of a matrix is a scalar value that can be used to solve linear equations and determine the invertibility of a matrix.

4

Subtraction of two matrices

Matrices can be subtracted from other matrices of the same size by subtracting the corresponding elements.

5

Transpose of a matrix

To transpose a matrix, just switch its rows and columns. It's like flipping a switch and seeing a new perspective.

Multiplication in Matrices (Source Code)

```
static int[][] multiply(int r1, int c1 ,int[][] arr1,int r2,int c2,int[][] arr2){
    int res[][] = new int[r1][c2];
    if( r2 != c1){
        System.out.println(x:"\n \tMatrix Multiplication is not possible ");
    }
    else{
        for(int i=0;i<r1;i++){
            for(int j=0;j<c2;j++){
                for(int k=0;k<c1;k++){
                    res[i][j] += arr1[i][k] * arr2[k][j];
                }
            }
        }
    }
    return res;
}
```

Matrix Multiplication in Java (Output)

```
        Enter the number of rows for 1st Matrix:
3

        Enter the number of columns for 1st Matrix:
3

        Enter elements of Matrix:
1 0 0
0 1 0
0 0 1

+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -

        Enter the number of rows for 2nd Matrix:
3

        Enter the number of columns for 2nd Matrix:
3

        Enter elements of Matrix:
11 22 33
44 55 66
77 88 99
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The 1st inputted matrix looks like:
1 0 0
0 1 0
0 0 1
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The 2nd inputted matrix looks like:
11 22 33
44 55 66
77 88 99
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -

        Multiplication of both the matrices:
11 22 33
44 55 66
77 88 99
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
```

Finding Transpose in Java

```java
static void transpose(int r1, int c1, int[][] arr1){
    System.out.println(x:"\n \tTranspose of the Given Matrix: ");
    for(int i =0;i<c1;i++){
        for(int j =0;j<r1;j++){
            System.out.print(arr1[j][i]+" ");
        }
        System.out.println();
    }
}
```

Source Code

```
        Enter the number of rows:
3

        Enter the number of columns:
3

        Enter elements of Matrix:
1 2 3
4 5 6
7 8 9
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The inputted matrix looks like:
1 2 3
4 5 6
7 8 9
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        Transpose of the Given Matrix:
1 4 7
2 5 8
3 6 9
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
```

Output

Addition of Matrices in Java

```java
static int[][] addition(int r1, int c1 ,int[][] arr1,int r2,int c2,int[][] arr2){
    int res[][] = new int[r1][c2];
    if( r2 != r1 && c2 != c1){
        System.out.println(x:"\n \tMatrix Addition is not possible ");
    }
    else{
        for(int i=0;i<r1;i++){
            for(int j=0;j<c2;j++){
                res[i][j] += arr1[i][j] + arr2[i][j];
            }
        }
    }
    return res;
}
```

Source Code

```
        Enter the number of rows for 1st Matrix:
3

        Enter the number of columns for 1st Matrix:
3

        Enter elements of Matrix:
1 2 3
4 5 6
7 8 9

+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -

        Enter the number of rows for 2nd Matrix:
3

        Enter the number of columns for 2nd Matrix:
3

        Enter elements of Matrix:
9 8 7
6 5 4
3 2 1
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The 1st inputted matrix looks like:
1 2 3
4 5 6
7 8 9
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The 2nd inputted matrix looks like:
9 8 7
6 5 4
3 2 1
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        Addition of both the matrices:
10 10 10
10 10 10
10 10 10
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
```

Output

Subtraction of Matrices in Java

```java
static int[][] substraction(int r1, int c1 ,int[][] arr1,int r2,int c2,int[][] arr2){
    int res[][] = new int[r1][c2];
    if( r2 != r1 && c2 != c1){
        System.out.println("\n \tMatrix Substraction is not possible ");
    }
    else{
        for(int i=0;i<r1;i++){
            for(int j=0;j<c2;j++){
                res[i][j] += arr1[i][j] - arr2[i][j];
            }
        }
    }
    return res;
}
```

Source Code.

```
        Enter the number of rows for 1st Matrix:
3

        Enter the number of columns for 1st Matrix:
3

        Enter elements of Matrix:
1 2 3
4 5 6
7 8 9

+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -

        Enter the number of rows for 2nd Matrix:
3

        Enter the number of columns for 2nd Matrix:
3

        Enter elements of Matrix:
1 0 0
0 1 0
0 0 1
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The 1st inputted matrix looks like:
1 2 3
4 5 6
7 8 9
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The 2nd inputted matrix looks like:
1 0 0
0 1 0
0 0 1
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The Substraction of both the matrices:
0 2 3
4 4 6
7 8 8
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
```

Output

Determinant of Matrices in Java

```java
static int determinantOfMatrix(int mat[][], int n)
{
    int D = 0; // Initialize result

    // Base case : if matrix
    // contains single element
    if (n == 1)
        return mat[0][0];

    // To store cofactors
    int temp[][] = new int[n][n];
    // To store sign multiplier
    int sign = 1;

    // Iterate for each element of first row
    for (int f = 0; f < n; f++) {
        // Getting Cofactor of mat[0][f]
        getCofactor(mat, temp, p:0, f, n);
        D += sign * mat[0][f]
            * determinantOfMatrix(temp, n - 1);

        // terms are to be added
        // with alternate sign
        sign = -sign;
    }

    return D;
}
```

Source Code

```
        Enter the number of rows:
2

        Enter the number of columns:
2

        Enter elements of Matrix:
1 2
3 4
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The inputted matrix looks like:
1 2
3 4
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
        The determinant of the given matrix: -2
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + -
```

Output

Conclusion

Matrices are a powerful tool in Java that can be used to perform complex mathematical operations, solve linear equations, and perform various types of transformations. We hope this presentation has inspired you to explore the use of matrices in your own Java projects!