DSA

Module 6

Matrix

Author
Srinivas Dande

Java Learning Center

6. Multi Dimensional Arrays

- Multi Dimensional Array is an Array of Arrays.

Single Dimensional Array	Collection of Actual value
Two Dimensional Array	Collection of Single Dimensional Arrays
Three Dimensional Array	Collection of 2-D Arrays

6.1. Two Dimensional Arrays

- Two Dimensional Arrays can be called as 2-D array.
- 2-D array is Array of 1-D Arrays.
- 2-D Array can be called as Matrix.
- **A matrix is a collection of numbers arranged into a fixed number of rows and columns.**

6.2. 2-D Array Declaration and Construction

Syntax:

<dataType> <refVarName>[][] = new <dataType>[<**size1**>][<**size2**>];
<dataType>[][] <refVarName> = new <dataType>[<**size1**>][<**size2**>];

Here

<**size1**> represents number of arrays to construct.
<**size2**> represents number of elements required for each array.

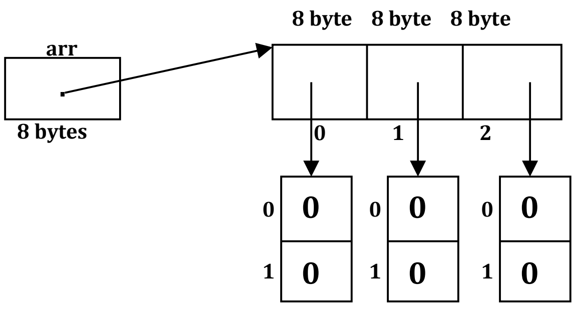
Ex:

```
int arr[][] = new int[3][2];
int[][] arr = new int[3][2];
```

6.3. 2-D Array Initialization

- **You can initialize array elements with your own value using index representation.**

int arr[][] = new int[3][2];

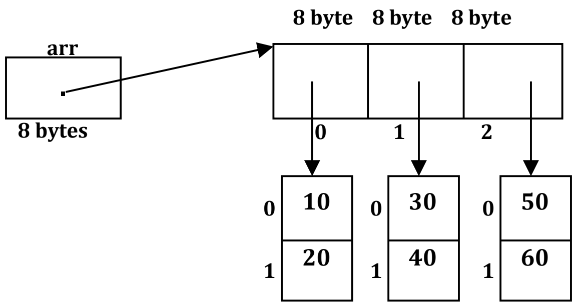


- **Here we are creating 3 arrays with size 2, and these three arrays addresses will be stored in main array.**

arr[0][0]	-> Represents the 1st element of 1st array which contains value 0.
arr[0][1]	-> Represents the 2nd element of 1st array which contains value 0.
arr[1][0]	-> Represents the 1st element of 2nd array which contains value 0.
arr[1][1]	-> Represents the 2nd element of 2nd array which contains value 0.
arr[2][0]	-> Represents the 1st element of 3rd array which contains value 0.
arr[2][1]	-> Represents the 2nd element of 3rd array which contains value 0.

- **You can initialize the array elements as follows:**

```
arr[0][0]    =10;
arr[0][1]    =20;
arr[1][0]    =30;
arr[1][1]    =40;
arr[2][0]    =50;
arr[2][1]    =60;
```



- **Accessing the elements after initialization**

arr[0][0] -> Represents the 1st element of 1st array which contains value 10.
arr[0][1] -> Represents the 2nd element of 1st array which contains value 20.
arr[1][0] -> Represents the 1st element of 2nd array which contains value 30.
arr[1][1] -> Represents the 2nd element of 2nd array which contains value 40.
arr[2][0] -> Represents the 1st element of 3rd array which contains value 50.
arr[2][1] -> Represents the 2nd element of 3rd array which contains value 60.

6.4. Jagged Arrays

Ex:

int arr[][] = new int[3][]; //**Second size is Optional**

arr[0]= new int[3];
arr[1]= new int[4];
arr[2]= new int[5];

- Now You can 12 Elements in the Array
- This Array is called as **Jagged Array.**

6.5. Array Declaration, Construction and Initialization

Syntax:

<dataType> <refVarName>[][] = {{v1,v2,...},{v1,v2,...},{v1,v2,...},...};

Ex:

int arr[][]={{10,20},{30,40},{50,60}};
int arr[][]={{10,20,30},{40},{50,60,70,80},{90,100}};

6.6. Examples on 2-D Arrays

Lab1.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab1 {
	public static void main(String[] args) {

		int arr[][] = new int[3][4];

		int rows = arr.length;
		System.out.println("Rows : "+rows);

		int r1size = arr[0].length;
		System.out.println("Row 1 Size : "+ r1size);

		int r2size = arr[1].length;
		System.out.println("Row 2 Size : "+ r2size);

		int r3size = arr[2].length;
		System.out.println("Row 3 Size : "+ r3size);

		for(int i=0;i<rows;i++) {
			System.out.println(arr[i]);
		}

	}
}
```

Lab2.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab2 {
        public static void main(String[] args) {

                //1. Creating 2-D Array
                int arr[][] = new int[3][3];

                //2. Accessing Elements of 2-D
                for(int i=0;i<arr.length;i++) {
                        for(int j=0;j<arr[i].length;j++) {
                                System.out.print(arr[i][j]+"\t");
                        }
                        System.out.println("");
                }

                //3.Initialyzing 1st Array/1st Row
                arr[0][0]=11;
                arr[0][1]=12;
                arr[0][2]=13;

                //4.Initialyzing 2nd Array/2nd Row
                arr[1][0]=21;
                arr[1][1]=22;
                arr[1][2]=23;

                //5.Initialyzing 3rd Array/3rd Row
                arr[2][0]=31;
                arr[2][1]=32;
                arr[2][2]=33;
                //6. Accessing Elements of 2-D
                for(int i=0;i<arr.length;i++) {
                        for(int j=0;j<arr[i].length;j++) {
                                System.out.print(arr[i][j]+"\t");
                        }
                        System.out.println("");
                }

        }
}
```

Lab3.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab3 {
        public static void main(String[] args) {
                // 1. Creating 2-D Jagged Array
                int arr[][] = new int[3][];

                // 2. Creating 1-D Arrays
                arr[0] = new int[3];
                arr[1] = new int[4];
                arr[2] = new int[5];

                // 3. Accessing Elements of 2-D
                for (int i = 0; i < arr.length; i++) {
                        for (int j = 0; j < arr[i].length; j++) {
                                System.out.print(arr[i][j] + "\t");
                        }
                        System.out.println("");
                }
                // 4.Initializing 2-D Array
                arr[0][0] = 11;
                arr[1][1] = 22;
                arr[2][2] = 33;

                arr[1][3] = 1;
                arr[2][3] = 1;
                arr[2][4] = 1;

                // 5. Accessing Elements of 2-D
                for (int i = 0; i < arr.length; i++) {
                        for (int j = 0; j < arr[i].length; j++) {
                                System.out.print(arr[i][j] + "\t");
                        }
                        System.out.println("");
                }
        }
}
```

Lab4.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab4 {
        public static void main(String[] args) {
                // 1. Creating and Initializing 2-D Array
                 int arr[][] = { {1,2,3},{4,5,6},{7,8,9} };

                // 2. Accessing Elements of 2-D
                for (int i = 0; i < arr.length; i++) {
                        for (int j = 0; j < arr[i].length; j++) {
                                System.out.print(arr[i][j] + "\t");
                        }
                        System.out.println("");
                }
        }
}
```

Lab5.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab5 {
	public static void main(String[] args) {

// 1. Creating and Initializing 2-D Array
int arr[][] = { { 1, 2, 3 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12, 13 }, { 14, 15, 16, 17, 18, 19 } };

		// 2. Accessing Elements of 2-D
		for (int i = 0; i < arr.length; i++) {
			for (int j = 0; j < arr[i].length; j++) {
				System.out.print(arr[i][j] + "\t");
			}
			System.out.println("");
		}

	}
}
```

Lab6.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab6 {
        public static void main(String[] args) {

                // 1. Creating and Initializing 2-D Array
                int arr[][] = { { 1, 2, 3, 4 },  { 5, 6, 7, 8 },   { 9, 10, 11, 12 },
                                        { 13, 14, 15, 16 } };

                System.out.println("1st Array --->");

                // 2. Accessing 1st Array Elements
                for (int j = 0; j < arr[0].length; j++) {
                                System.out.print(arr[0][j] + "\t");
                }

                System.out.println("\n \n 2nd Array --->");

                // 3. Accessing 2nd Array Elements
                for (int j = 0; j < arr[1].length; j++) {
                                System.out.print(arr[1][j] + "\t");
                }

                 System.out.println("\n \n 3rd Array --->");

                // 4. Accessing 3rd Array Elements
                        for (int j = 0; j < arr[2].length; j++) {
                                        System.out.print(arr[2][j] + "\t");
                        }

                }
        }
```

6.7. Problems on Matrices

Problem1A.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem1- Print the Main Diagonal of n * n Matrix

public class Problem1A {
        public static void main(String[] args) {

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

                //Main Diagonal
                for (int i = 0; i < mat.length; i++) {
                        for (int j = 0; j < mat[i].length; j++) {
                                if (i == j)
                                        System.out.print(mat[i][j] + "\t");
                        }
                }

        }
}

// Time Complexity - -  O(n^2)
// Aux Space - O(1)
```

Problem1B.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem1- Print the Main Diagonal of n * n Matrix

public class Problem1B {
        public static void main(String[] args) {

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

                //Main Diagonal
                for (int i = 0; i < mat.length; i++) {
                                System.out.print(mat[i][i] + "\t");
                }

        }
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem2.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem2- Print the Secondary Diagonal of n * n Matrix

public class Problem2 {
	public static void main(String[] args) {

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

		int n = mat.length;

		//Main Diagonal
		for (int i = 0,j=n-1; i < n; i++,j--) {
			System.out.print(mat[i][j] + "\t");
		}

	}
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem3A.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem3- Sum of Diagonal elements of n * n Matrix

public class Problem3A {
	public static void main(String[] args) {

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

		int n = mat.length;
		int sum = 0;

		// Main Diagonal Sum
		for (int i = 0; i < n; i++) {
			sum = sum + mat[i][i];
		}

		// Secondary Diagonal Sum
		for (int i = 0, j = n - 1; i < n; i++, j--) {
			if (i != j)
				sum = sum + mat[i][j];
		}

		System.out.println("Sum : " + sum);
	}
}

// Time Complexity - -  O(2n) /O(n)
// Aux Space - O(1)
```

Problem3B.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem3- Sum of Diagonal elements of n * n Matrix

public class Problem3B {
	public static void main(String[] args) {

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

		int n = mat.length;
		int sum = 0;

		for (int i = 0, j = n - 1; i < n; i++, j--) {
			if (i != j)
				sum = sum + mat[i][i]+ mat[i][j];
			else
				sum = sum + mat[i][i];
		}

		System.out.println("Sum : " + sum);
	}
}

// Time Complexity - -  O(n)
// Aux Space - O(1)
```

Problem4.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Problem4- Reverse the Columns of Matrix

public class Problem4 {

	static void printMatrix(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] + "\t");
			}
			System.out.println("");
		}
	}

	static void reverseColums(int mat[][]) {
		int n = mat.length;

		for (int i = 0; i < n; i++) {
			int start = 0;
			int end = n - 1;
			while (start < end) {
				int temp = mat[start][i];
				mat[start][i] = mat[end][i];
				mat[end][i] = temp;

				start++;
				end--;
			}
		}
	}
```

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

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

                printMatrix(mat);
                System.out.println("--------------------");
                reverseColums(mat);
                printMatrix(mat);
        }
}
// Time Complexity - -  O(n^2)
// Aux Space - O(1)
```

Problem5.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem5- Reverse the Rows of Matrix

public class Problem5 {

        static void printMatrix(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] + "\t");
                        }
                        System.out.println("");
                }
        }
        static void reverseRows(int mat[][]) {
            int n = mat.length;

            for (int i = 0; i < n; i++) {
                int start = 0;
                int end = n - 1;
                while (start < end) {
                    int temp = mat[i][start];
                    mat[i][start] = mat[i][end];
                    mat[i][end] = temp;

                    start++;
                    end--;
                }

            }
    }

    public static void main(String[] args) {

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

            printMatrix(mat);
            System.out.println("--------------------");
            reverseRows(mat);
            printMatrix(mat);
    }
}
```

// Time Complexity - - O(n^2)
// Aux Space - O(1)

Problem6.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem6- Transpose of Matrix
// Swap the Rows to Cols and Cols to Rows

public class Problem6 {

	static void printMatrix(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] + "\t");
			}
			System.out.println("");
		}
	}

	static void transpose(int mat[][]) {

		int n = mat.length;

		for (int i = 0; i < n; i++) {
			for (int j = i + 1; j < n; j++) {
				int temp = mat[i][j];
				mat[i][j] = mat[j][i];
				mat[j][i] = temp;
			}

		}
	}
```

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

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

                printMatrix(mat);
                System.out.println("-------------------");
                transpose(mat);
                printMatrix(mat);
        }
}

// Time Complexity - -  O(n^2)
// Aux Space - O(1)
```

Problem7.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

//Problem7- Rotate of Matrix to Anti-Clockwise by 90 degrees.

public class Problem7 {

        static void printMatrix(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] + "\t");
                        }
                        System.out.println("");
                }
        }
```

```
static void transpose(int mat[][]) {
        int n = mat.length;

        for (int i = 0; i < n; i++) {
                for (int j = i + 1; j < n; j++) {

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

        }
}

static void reverseColums(int mat[][]) {
        int n = mat.length;

        for (int i = 0; i < n; i++) {
                int start = 0;
                int end = n - 1;
                while (start < end) {
                        int temp = mat[start][i];
                        mat[start][i] = mat[end][i];
                        mat[end][i] = temp;

                        start++;
                        end--;
                }

        }
}

static void rotateLeft90(int mat[][]) {
        transpose(mat);
        reverseColums(mat);
}
```

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

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

		printMatrix(mat);
		System.out.println("-------------------");
		rotateLeft90(mat);
		//rotateLeft90(mat);
		//rotateLeft90(mat);
		printMatrix(mat);
	}
}

// Time Complexity - -  O(n^2)
// Aux Space - O(1)
```

Problem8.java

```
package com.jlcindia.matrices;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Problem8- Rotate of Matrix to Clockwise by 90 degrees.

public class Problem8 {
	static void printMatrix(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] + "\t");
			}
			System.out.println("");
		}
	}
```

```
static void transpose(int mat[][]) {
        int n = mat.length;

        for (int i = 0; i < n; i++) {
                for (int j = i + 1; j < n; j++) {
                        int temp = mat[i][j];
                        mat[i][j] = mat[j][i];
                        mat[j][i] = temp;
                }
        }
}

static void reverseRows(int mat[][]) {
        int n = mat.length;

        for (int i = 0; i < n; i++) {
                int start = 0;
                int end = n - 1;
                while (start < end) {
                        int temp = mat[i][start];
                        mat[i][start] = mat[i][end];
                        mat[i][end] = temp;

                        start++;
                        end--;
                }
        }
}

static void rotateRight90(int mat[][]) {
        transpose(mat);
        reverseRows(mat);
}
```

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

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

                printMatrix(mat);
                System.out.println("--------------------");
                rotateRight90(mat);
                //rotateRight90(mat);
                //rotateRight90(mat);
                printMatrix(mat);

        }
}

// Time Complexity - -  O(n^2)
// Aux Space - O(1)
```