Arrays in JAVA

An array is a collection of similar data types. Array is a container object that hold values of homogeneous type. It is also known as static data structure because size of an array must be specified at the time of its declaration.

Array starts from zero index and goes to n-1 where n is length of the array.

In Java, array is treated as an object and stores into heap memory. It allows to store primitive values or reference values.

Array can be single dimensional or multidimensional in Java.

- Features of Array

It is always indexed. Index begins from 0.

It is a collection of similar data types.

It occupies a contiguous memory location.

It allows to access elements randomly.

- Single Dimensional Array

Single dimensional array use single index to store elements. You can get all the elements of array by just increment its index by one.

Array Declaration

Syntax :

datatype[] arrayName;;

or

datatype arrayName[];

Java allows to declare array by using both declaration syntax, both are valid.

The arrayName can be any valid array name and datatype can be any like: int, float, byte etc.

Example:

int[ ] arr;

char[ ] arr;

short[ ] arr;

long[ ] arr;

int[ ][ ] arr;   // two dimensional array.

Initialization of Array

Initialization is a process of allocating memory to an array. At the time of initialization, we specify the size of array to reserve memory area.

Initialization Syntax:

```
 arrayName = new datatype[size]
```

new operator is used to initialize an array.

The arrayName is the name of array, new is a keyword used to allocate memory and size is length of array.

We can combine both declaration and initialization in a single statement.

```
Datatype[] arrayName = new datatype[size]
```

Example: Create an Array

Let's create a single dimensional array.

```
class Demo
{
public static void main(String[] args)
  {
    int[] arr = new int[5];
        for(int x : arr)
        {
                System.out.println(x);
        }
   }
}
```

In the above example, we created an array arr of int type and can store 5 elements. We iterate the array to access its elements and it prints five times zero to the console. It prints zero because we did not set values to array, so all the elements of the array initialized to 0 by default.

Set Array Elements

We can set array elements either at the time of initialization or by assigning direct to its index.

```
int[] arr = {10,20,30,40,50};
```

Here, we are assigning values at the time of array creation. It is useful when we want to store static data into the array.

or

```
arr[1] = 105
```

Here, we are assigning a value to array’s 1 index. It is useful when we want to store dynamic data into the array.

Array Example

Here, we are assigning values to array by using both the way discussed above.

```
class Demo
{
public static void main(String[] args)
 {
        int[] arr = {10,20,30,40,50};
      for(int x : arr)
      {
            System.out.println(x);
      }
   // assigning a value
      arr[1] = 105;
      System.out.println("element at first index: " +arr[1]);
  }
}
```

Accessing array element

we can access array elements by its index value. Either by using loop or direct index value. We can use loop like: for, for-each or while to traverse the array elements.

Example to access elements

```
class Demo

{

public static void main(String[] args)

 {

        int[] arr = {10,20,30,40,50};

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

      {

            System.out.println(arr[i]);

      }

      System.out.println("element at first index: " +arr[1]);

  }

}
```

Here, we are traversing array elements using loop and accessed an element randomly.

Note:-To find the length of an array, we can use length property of array object like: array_name.length.

Multi-Dimensional Array

A multi-dimensional array is very much similar to a single dimensional array. It can have multiple rows and multiple columns unlike single dimensional array, which can have only one row index.

It represent data into tabular form in which data is stored into row and columns.

Multi-Dimensional Array Declaration

datatype[ ][ ] arrayName;

Initialization of Array

datatype[ ][ ] arrayName = new int[no_of_rows][no_of_columns];

The arrayName is the name of array, new is a keyword used to allocate memory and no_of_rows and no_of_columns both are used to set size of rows and columns elements.

Like single dimensional array, we can statically initialize multi-dimensional array as well.

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

Example:

```
class Demo
{
public static void main(String[] args)
  {
          int arr[ ][ ] = {{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15}};
       for(int i=0;i<3;i++)
       {
          for (int j = 0; j < 5; j++) {
                    System.out.print(arr[i][j]+" ");
                              }
          System.out.println();


       }
    // assigning a value
       System.out.println("element at first row and second column: " +arr[0][1]);
   }
}
```

Jagged Array

Jagged array is an array that has different numbers of columns elements. In java, a jagged array means to have a multi-dimensional array with uneven size of columns in it.

Initialization of Jagged Array

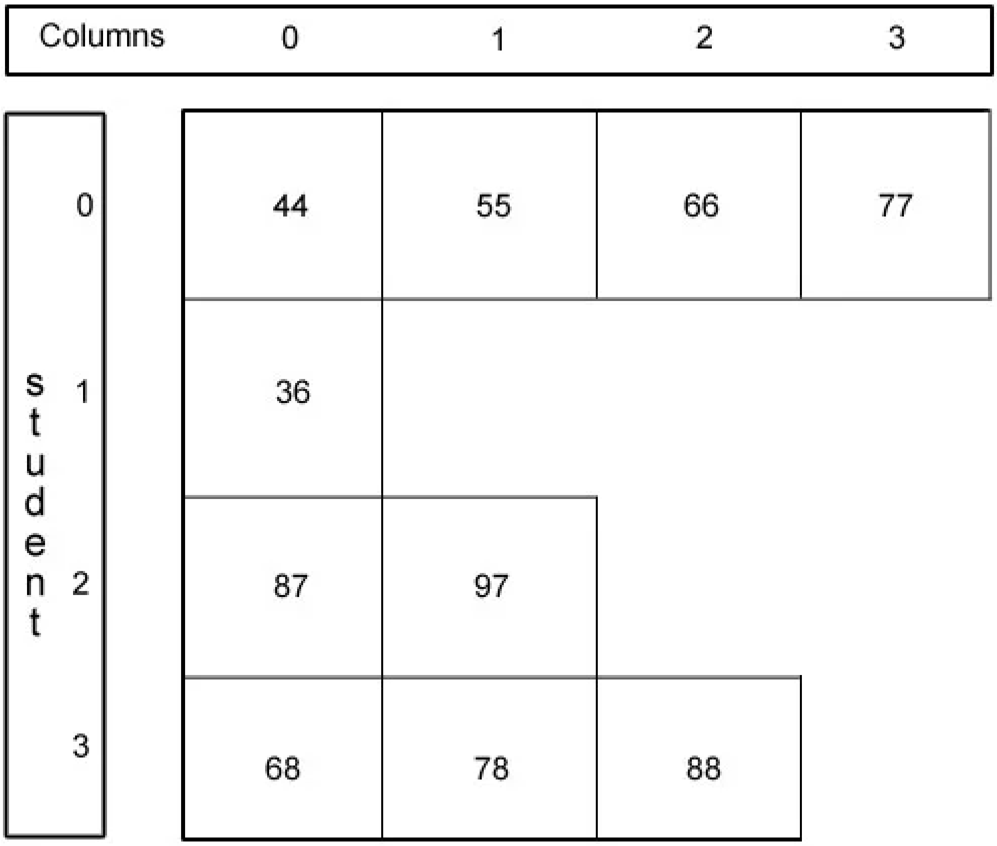
Jagged array initialization is different little different. We have to set columns size for each row independently.

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

arr[0] = new int[3];

arr[1] = new int[4];

arr[2] = new int[5];
```



Columns	0	1	2	3
student 0	44	55	66	77
1	36			
2	87	97		
3	68	78	88	

FIgure : Varying columns 2D array - matrix form

- Example : Jagged Array

```
   class Demo
{
public static void main(String[] args)
 {
   int arr[ ][ ] = {{1,2,3},{4,5},{6,7,8,9}};
  for(int i=0;i<3;i++)
  {
   for (int j = 0; j < arr[i].length; j++) {
      System.out.print(arr[i][j]+" ");
         }
   System.out.println();
  }
 }
}
```

Here, we can see number of rows are 3 and columns are different for each row. This type of array is called jagged array.