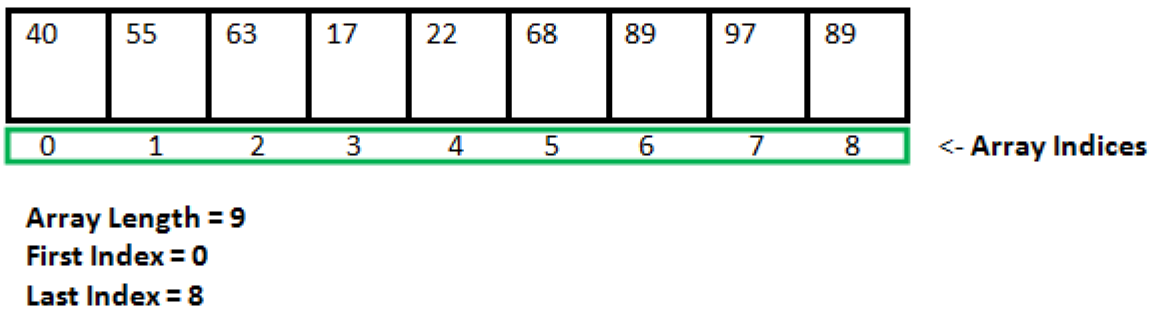
What is an Array and explain different types of arrays in java

Arrays:

An array is a collection of similar type of elements which has contiguous memory location. **Java array** is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.



type var-name[];

OR

type[] var-name;

Disadvantages:

Size Limit:

We can store only the fixed size of elements in the array. It doesn't grow its size at runtime.

Types of Array in java

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Single Dimensional Array/ One Dimensional:

- One Dimensional Array in java is always used with only **one subscript ([ ])**.
- One Dimensional Array in java is always used with only one subscript ([]).
- A one-dimensional array behaves likes a list of variables.
- You can access the variables of an array by using an index in square brackets preceded by the name of that array.
- Index value should be an integer. Before using the array, we must declare it.

```
type var-name[];
OR
type[] var-name;
```

Example Program:1

```
class OnedimenisionaArray
 {
   public static void main(String[] args)
   {
      int[] arr;  // declares an Array of integers.
      arr = new int[5];  // allocating memory for 5 integers.
      arr[0] = 10; // initialize the first elements of the array
      arr[1] = 20;   // initialize the second elements of the array
      arr[2] = 30;
      arr[3] = 40;
      arr[4] = 50;
     // accessing the elements of the specified array
       for (int i = 0; i < arr.length; i++)
       System.out.println("Element at index " + i + " : " + arr[i]);
   }
}
```

Output:

```
Element at index 0 : 10

Element at index 1 : 20

Element at index 2 : 30

Element at index 3 : 40

Element at index 4 : 50
```

Multidimensional Arrays:

Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

data_type[1st dimension][2nd dimension][]..[Nth dimension] **array_name** = **new data_type**[size1][size2]….[sizeN];

- **data_type**: Type of data to be stored in the array. For example: int, char, etc.
- **dimension**: The dimension of the array created. For example: 1D, 2D, etc.
- **array_name**: Name of the array
- **size1, size2, …, sizeN**: Sizes of the dimensions respectively.

Two dimensional array:

2D array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. The 2d array represents with two sub scripts.

Initializing 2D Arrays:

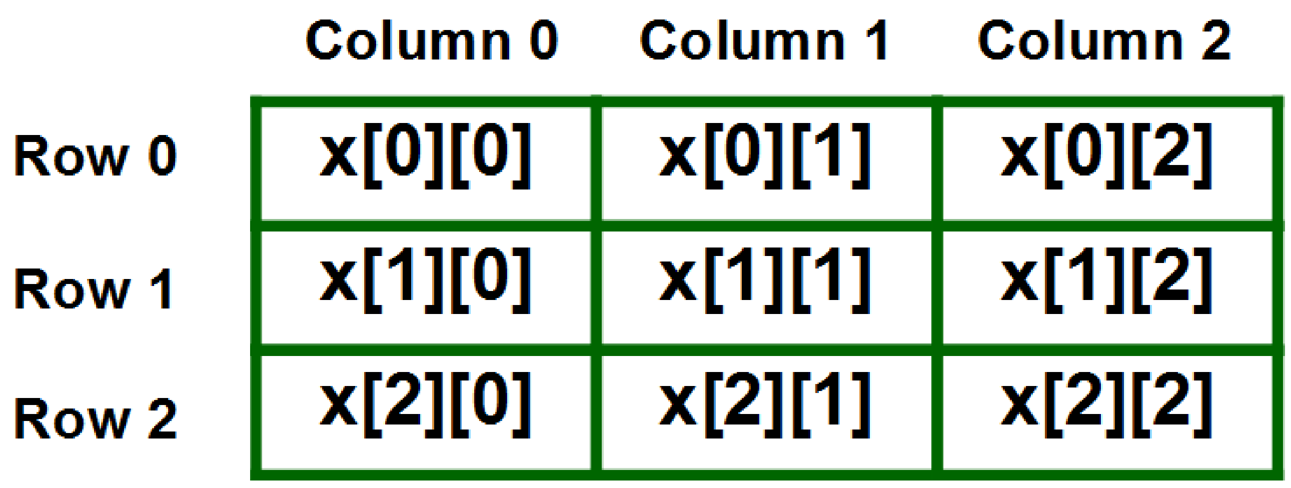
```
int arr[2][2] = {0,1,2,3};

int[][] twoD_arr = new int[10][20];
```

Two – dimensional Array (2D-Array)
Two – dimensional array is the simplest form of a multidimensional array. A two – dimensional array can be seen as an array of one – dimensional array for easier understanding.

Representation of 2D array in Tabular Format:

A two – dimensional array can be seen as a table with ‘x’ rows and ‘y’ columns where the row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1). A two – dimensional array ‘x’ with 3 rows and 3 columns is shown below:



	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Example:2

```
public class TwoDimensional
{
public static void main(String args[])
{
// declaring and initializing 2D array
int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };
// printing 2D array
for (int i = 0; i < 3; i++)
{
for (int j = 0; j < 3; j++)
System.out.print(arr[i][j] + " ");
System.out.println();
    }
  }
}
```

Output:

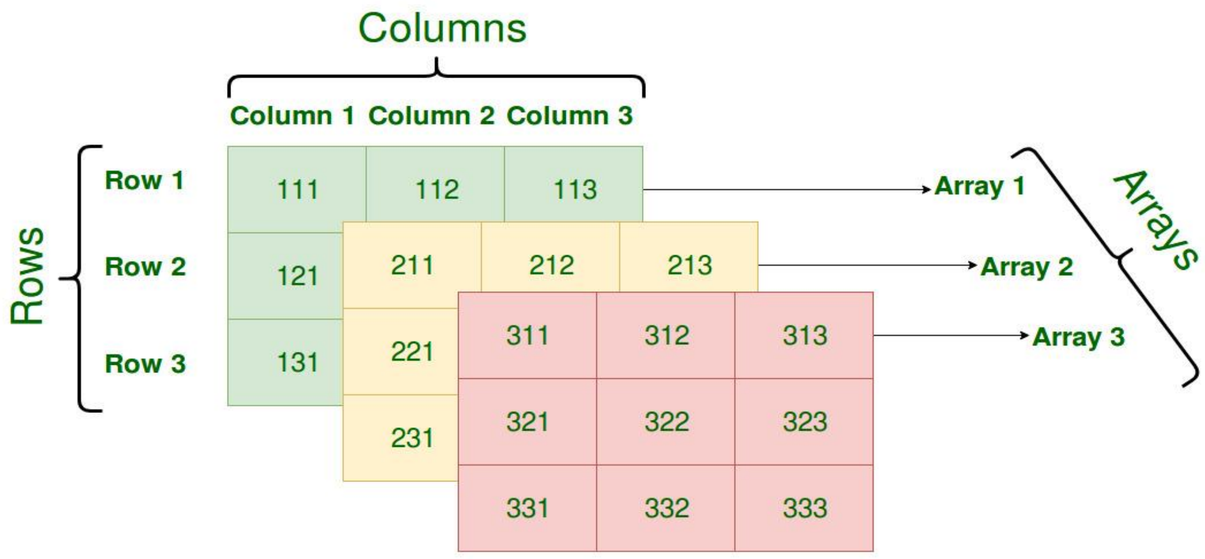
```
2 7 9

3 6 1

7 4 2
```

Multidimensional array:

Three – dimensional array is a complex form of a multidimensional array. A three – dimensional array can be seen as an array of two – dimensional array for easier understanding. Multi-dimensional array represents with three subscripts. Elements in three-dimensional arrays are commonly referred by **x[i][j][k]** where ‘i’ is the array number, ‘j’ is the row number and ‘k’ is the column number.



Example:3

```
class MultiDimensional
 {
   public static void main(String[] args)
   {
       int[][][] arr = { { { 1, 2 }, { 3, 4 } },
                         { { 5, 6 }, { 7, 8 } } };

      for (int i = 0; i < 2; i++)
        {
         for (int j = 0; j < 2; j++)
          {
            for (int k = 0; k < 2; k++)
              {
                System.out.print(arr[i][j][k] + " ");
             }
            System.out.println();
         }
        System.out.println();
      }
   }
}
```

Output:

```
1 2

3 4

5 6

7 8
```

Practice Programs:1

```
public class ReverseArray
{
public static void main(String[] args)
{
 //Initialize array
int [] arr = new int [] {1, 2, 3, 4, 5};
System.out.println("Original array: ");
for (int i = 0; i < arr.length; i++)
{
System.out.print(arr[i] + " ");
}
System.out.println();
System.out.println("Array in reverse order: ");
//Loop through the array in reverse order
for (int i = arr.length-1; i >= 0; i--)
{
System.out.print(arr[i] + " ");
}
}
}
```

Output:

```
Original array: 1 2 3 4 5

Array in reverse order:

5 4 3 2 1
```

Example:2

```
public class EvenPosition
{
public static void main(String[] args)
{
//Initialize array
int [] arr = new int [] {1, 2, 3, 4, 5};
System.out.println("Elements of given array present on even position: ");
//Loop through the array by incrementing value of i by 2
//Here, i will start from 1 as first even positioned element is present at position 1. for (int
i = 1; i < arr.length; i = i+2)
{
System.out.println(arr[i]);
}
}
}
```

Output:

```
Elements of given array present on even position:
2
4
```

Example:3

```
public class OddPosition
{
public static void main(String[] args)
{
//Initialize array
int [] arr = new int [] {1, 2, 3, 4, 5};
System.out.println("Elements of given array present on odd position: ");
//Loop through the array by incrementing value of i by 2
for (int i = 0; i < arr.length; i = i+2)
{
System.out.println(arr[i]);
}
}
}
```

Output:

```
Elements of given array present on odd position:
1
3
5
```

Example:4

```
public class OddEvenInArrayExample
{
public static void main(String args[])
{
int a[]={1,2,5,6,3,2};
System.out.println("Odd Numbers:");
for(int i=0;i<a.length;i++)
{
if(a[i]%2!=0)
{
System.out.println(a[i]);
}
}
System.out.println("Even Numbers:");
for(int i=0;i<a.length;i++)
{
if(a[i]%2==0)
{
System.out.println(a[i]);
}
}
}
}
```

Output:

```
Odd Numbers:
1
5
3
Even Numbers:
2
6
2
```

Example: add two matrices

```
public class MatrixAdditionExample
{
public static void main(String args[])
{
//creating two matrices
int a[][]={{1,3,4},{2,4,3},{3,4,5}};
int b[][]={{1,3,4},{2,4,3},{1,2,4}};

//creating another matrix to store the sum of two matrices
int c[][]=new int[3][3];  //3 rows and 3 columns

//adding and printing addition of 2 matrices
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
c[i][j]=a[i][j]+b[i][j];    //use - for subtraction
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}
}
}
```

Output:

```
2 6 8
4 8 6
4 6 9
```

Example: multiply two matrices

```
public class MatrixMultiplicationExample
{
public static void main(String args[])
{
//creating two matrices
int a[][]={{1,1,1},{2,2,2},{3,3,3}};
int b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two matrices
int c[][]=new int[3][3];  //3 rows and 3 columns

//multiplying and printing multiplication of 2 matrices
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
c[i][j]=0;
for(int k=0;k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}//end of k loop
System.out.print(c[i][j]+" ");  //printing matrix element
}//end of j loop
System.out.println();//new line
}
}
}
```

Output:

```
6    6   6
12 12 12
18 18 18
```

Example: Transpose matrix

```
public class MatrixTransposeExample
{
public static void main(String args[])
{
//creating a matrix
int original[][]={{1,3,4},{2,4,3},{3,4,5}};

//creating another matrix to store transpose of a matrix
int transpose[][]=new int[3][3];  //3 rows and 3 columns

//Code to transpose a matrix
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
transpose[i][j]=original[j][i];
}
}

System.out.println("Printing Matrix without transpose:");
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
System.out.print(original[i][j]+" ");
}
System.out.println();//new line
}
System.out.println("Printing Matrix After Transpose:");
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
System.out.print(transpose[i][j]+" ");
}
System.out.println(); //new line
}
}
}
```

Output:

```
Printing Matrix without transpose:
1 3 4
2 4 3
3 4 5
Printing Matrix After Transpose:
1 2 3
3 4 4
4 3 5
```