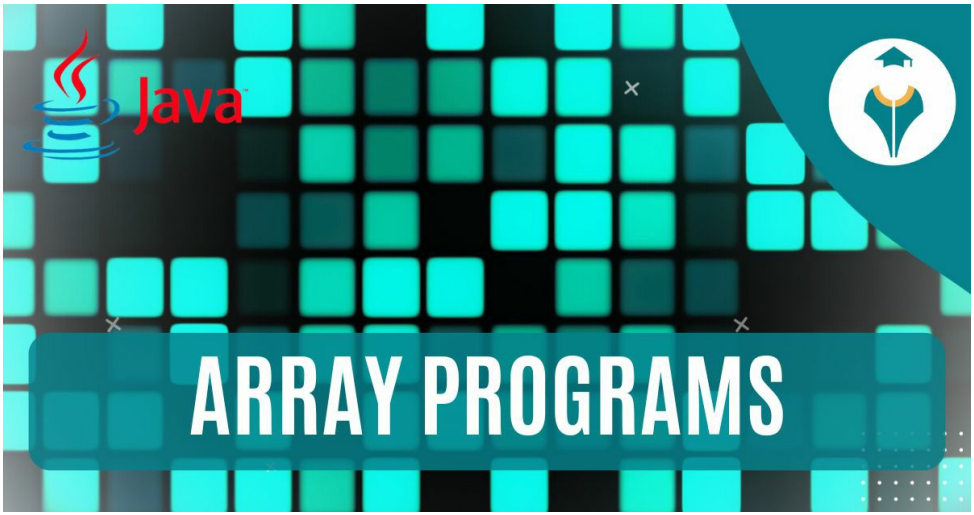
Array Programs in Java | Beginner to Expert Level

Esha Gupta
Associate Senior Executive

Updated on Mar 21, 2024 17:10 IST

Array programs in Java traverse from basic single-dimensional arrays to complex multi-dimensional arrays and dynamic arrays using ArrayList. From initializing and accessing array elements to advanced operations like sorting and searching, arrays facilitate efficient data management and manipulation, forming a fundamental concept for every aspiring Java developer. Let's understand more!



An **Array Program** typically refers to a program that utilizes array data structures that can hold more than one value at a time. Arrays are used in programming to organize a series of items sequentially.

Read more: Arrays in Java.

This blog will help you learn Array Programs in Java from Beginner (Level 1) to Expert (Level 4).

Table of Content

Level 1 (Beginners)

Here, we will see some very basic array programs!

Question 1: Creating and Initializing an Array

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] myArray = {1, 2, 3, 4, 5};
        System.out.println("Array created with elements: " +
java.util.Arrays.toString(myArray));
    }
}
```

Output

Array created with elements: [1, 2, 3, 4, 5]

Question 2: Accessing Elements of an Array

```java
public class Main {
    public static void main(String[] args) {
        int[] myArray = {1, 2, 3, 4, 5};
        System.out.println("The first element is: " + myArray[1]);
        System.out.println("The fourth element is: " + myArray[4]);
    }
}
```

Output

The first element is: 2
The fourth element is: 5

Question 3: Modifying Elements of an Array

```java
public class Main {
    public static void main(String[] args) {
        int[] myArray = {1, 2, 3, 4, 5};
        myArray[1] = 10;
        System.out.println("Array after modification: " + java.util.Arrays.toString(myArray));
    }
}
```

Output

Array after modification: [1, 10, 3, 4, 5]

Question 4: Finding the Length of an Array

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] myArray = {1, 2, 3, 4, 6};
        System.out.println("The length of the array is: " + myArray.length);
    }
}
```

Output

The length of the array is: 5

Question 5: Looping through an Array

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] myArray = {1, 2, 3, 4, 5};
        for (int i = 0; i < myArray.length; i++) {
            System.out.println("Element at index " + i + ": " + myArray[i]);
        }
    }
}
```

Output

Element at index 0: 1

Element at index 1: 2
Element at index 2: 3
Element at index 3: 4
Element at index 4: 5

Question 6: Multi-dimensional Arrays

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[][] multiArray = { {1, 2}, {3, 4}, {5, 6} };
        System.out.println("Element at row 1 column 1: " + multiArray[0][0]);
    }
}
```

Output

Element at row 1 column 1: 1

Question 7: Copying Arrays

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] myArray = {1, 2, 3, 4, 5};
        int[] newArray = java.util.Arrays.copyOf(myArray, myArray.length);
        System.out.println("Copied array: " + java.util.Arrays.toString(newArray));
    }
}
```

Output

Copied array: [1, 2, 3, 4, 5]

Question 8: Sorting Arrays

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] myArray = {5, 3, 4, 1, 2};
        java.util.Arrays.sort(myArray);
        System.out.println("Sorted array: " + java.util.Arrays.toString(myArray));
    }
}
```

Output

Sorted array: [1, 2, 3, 4, 5]

Level 2 (Intermediate Level)

Question 1: Find the Second Largest Number in an Array

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6};
        int largest = Integer.MIN_VALUE;
        int secondLargest = Integer.MIN_VALUE;

        for (int num : arr) {
            if (num > largest) {
                secondLargest = largest;
                largest = num;
            } else if (num > secondLargest && num != largest) {
                secondLargest = num;
            }
        }

        System.out.println("The second largest number is: " + secondLargest);
    }
}
```

Output

The second largest number is: 5

Question 2: Reverse an Array

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int n = arr.length;

        for (int i = 0; i < n / 2; i++) {
            int temp = arr[i];
            arr[i] = arr[n - i - 1];
            arr[n - i - 1] = temp;
        }

        System.out.println("Reversed array: " + java.util.Arrays.toString(arr));
    }
}
```

Output

Reversed array: [5, 4, 3, 2, 1]

Question 3: Find the Duplicate Elements

Copy code

```
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 2, 5, 6, 3};
        Set<Integer> set = new HashSet<>();
        Set<Integer> duplicates = new HashSet<>();

        for (int num : arr) {
            if (!set.add(num)) {
                duplicates.add(num);
            }
        }

        System.out.println("Duplicate elements: " + duplicates);
    }
}
```

Output

Duplicate elements: [2, 3]

Question 4: Find the Kth Largest and Smallest Element in an Array

```java
public class Main {
    public static void main(String[] args) {
        int[] arr = {7, 5, 9, 3, 2, 8, 1, 6};
        int k = 5;
        java.util.Arrays.sort(arr);
        System.out.println(k + "th largest element: " + arr[arr.length - k]);
        System.out.println(k + "th smallest element: " + arr[k - 1]);
    }
}
```

Copy code

Output

5th largest element: 5
5th smallest element: 6

Question 5: Move all Zeroes to the End of the Array

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 0, 2, 0, 3, 0, 4, 0};
        int n = arr.length;
        int count = 0;

        for (int i = 0; i < n; i++) {
            if (arr[i] != 0) {
                arr[count++] = arr[i];
            }
        }
        while (count < n) {
            arr[count++] = 0;
        }

        System.out.println("Array after moving zeroes: " + java.util.Arrays.toString(arr));
    }
}
```

Output

Array after moving zeroes: [1, 2, 3, 4, 0, 0, 0, 0]

Question 6: Find the “Leaders” in an Array

```java
public class Main {
    public static void main(String[] args) {
        int[] arr = {16, 17, 4, 3, 5, 2};
        int n = arr.length;
        int maxFromRight = arr[n - 1];

        System.out.print("Leaders: " + maxFromRight + " ");

        for (int i = n - 2; i >= 0; i--) {
            if (arr[i] > maxFromRight) {
                maxFromRight = arr[i];
                System.out.print(maxFromRight + " ");
            }
        }
    }
}
```

Copy code

Output

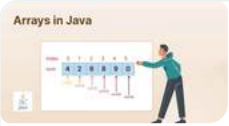
Leaders: 2 5 17

A leader in an array is an element which is larger than all the elements to its right side, which is what you can see in the above output.

Question 7: Find the Majority Element

Understanding ArrayList in Java

The below article goes through explaining ArrayList in Java with suitable examples. It covers the creation and operations on ArrayList along with a few methods in it. Let's begin!



Implementing Array in Java

Arrays in Java help to maintain elements of same datatype under a single variable name. The below article explains the implementation of array in java with examples.

Understanding Data Structures and Algorithms in Java

Data Structure in Java is used to store and organize data efficiently while the algorithms are used to manipulate the data in that structure. In this article, we will briefly...**read more**

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {3, 3, 4, 2, 4, 4, 2, 4, 4};
        int candidate = findCandidate(arr);
        if (isMajority(arr, candidate)) {
            System.out.println("Majority element is: " + candidate);
        } else {
            System.out.println("No majority element found");
        }
    }

    static int findCandidate(int[] arr) {
        int majIndex = 0, count = 1;
        for (int i = 1; i < arr.length; i++) {
            if (arr[majIndex] == arr[i]) {
                count++;
            } else {
                count--;
            }
            if (count == 0) {
```

```
                majIndex = i;
                count = 1;
            }
        }
        return arr[majIndex];
    }

    static boolean isMajority(int[] arr, int candidate) {
        int count = 0;
        for (int num : arr) {
            if (num == candidate) {
                count++;
            }
        }
        return count > arr.length / 2;
    }
}
```

Output

Majority element is: 4

A majority element in an array is an element that appears more than n/2 times where n is the size of the array, which is what you can see in the above output.

Question 8: Rotate Array by N Elements

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6, 7};
        int n = 2; // Number of positions to rotate
        reverseArray(arr, 0, n - 1);
        reverseArray(arr, n, arr.length - 1);
        reverseArray(arr, 0, arr.length - 1);

        System.out.println("Rotated array: " + java.util.Arrays.toString(arr));
    }

    static void reverseArray(int[] arr, int start, int end) {
        while (start < end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }
}
```

Output

Rotated array: [3, 4, 5, 6, 7, 1, 2]

Level 3 (Advanced Level)

Question 1: Maximum Subarray Sum (Kadane's Algorithm)

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] nums = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
        System.out.println(maxSubArray(nums));
    }

    public static int maxSubArray(int[] nums) {
        int maxSoFar = nums[0], maxEndingHere = nums[0];
        for (int i = 1; i < nums.length; i++) {
            maxEndingHere = Math.max(maxEndingHere + nums[i], nums[i]);
            maxSoFar = Math.max(maxSoFar, maxEndingHere);
        }
        return maxSoFar;
    }
}
```

Output

6

Question 2: Longest Increasing Subsequence

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] nums = {10, 22, 9, 33, 21, 50, 41, 60};
        System.out.println(lengthOfLIS(nums));
    }

    public static int lengthOfLIS(int[] nums) {
        int[] dp = new int[nums.length];
        int length = 0;

        for (int num : nums) {
            int i = java.util.Arrays.binarySearch(dp, 0, length, num);
            if (i < 0) i = -(i + 1);
            dp[i] = num;
            if (i == length) length++;
        }
        return length;
    }
}
```

Output

5

Question 3: Matrix Spiral Traversal

Copy code

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

```
    int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    System.out.println(spiralOrder(matrix));
}

public static java.util.List<Integer> spiralOrder(int[][] matrix) {
    java.util.List<Integer> res = new java.util.ArrayList<>();
    if (matrix.length == 0) return res;
    int rowBegin = 0, rowEnd = matrix.length - 1;
    int colBegin = 0, colEnd = matrix[0].length - 1;

    while (rowBegin <= rowEnd && colBegin <= colEnd) {
        for (int j = colBegin; j <= colEnd; j++) {
            res.add(matrix[rowBegin][j]);
        }
        rowBegin++;
        for (int j = rowBegin; j <= rowEnd; j++) {
            res.add(matrix[j][colEnd]);
        }
        colEnd--;
        if (rowBegin <= rowEnd) {
            for (int j = colEnd; j >= colBegin; j--) {
                res.add(matrix[rowEnd][j]);
            }
        }
        rowEnd--;
        if (colBegin <= colEnd) {
            for (int j = rowEnd; j >= rowBegin; j--) {
                res.add(matrix[j][colBegin]);
            }
        }
        colBegin++;
    }
    return res;
```

```
    }
}
```

Output

[1, 2, 3, 6, 9, 8, 7, 4, 5]

Question 4: Median of Two Sorted Arrays

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] nums1 = {1, 3};
        int[] nums2 = {2};
        System.out.println(findMedianSortedArrays(nums1, nums2));
    }

    public static double findMedianSortedArrays(int[] nums1, int[] nums2) {
        if (nums1.length > nums2.length) {
            int[] temp = nums1;
            nums1 = nums2;
            nums2 = temp;
        }
        int m = nums1.length, n = nums2.length;
        int imin = 0, imax = m, halfLen = (m + n + 1) / 2;
        double median = 0.0;

        while (imin <= imax) {
            int i = (imin + imax) / 2;
            int j = halfLen - i;

            if (i < m && nums2[j - 1] > nums1[i]) {
```

```
            imin = i + 1;
        } else if (i > 0 && nums1[i - 1] > nums2[j]) {
            imax = i - 1;
        } else {
            int maxOfLeft = 0;
            if (i == 0) maxOfLeft = nums2[j - 1];
            else if (j == 0) maxOfLeft = nums1[i - 1];
            else maxOfLeft = Math.max(nums1[i - 1], nums2[j - 1]);

            if ((m + n) % 2 == 1) return maxOfLeft;

            int minOfRight = 0;
            if (i == m) minOfRight = nums2[j];
            else if (j == n) minOfRight = nums1[i];
            else minOfRight = Math.min(nums1[i], nums2[j]);

            return (maxOfLeft + minOfRight) / 2.0;
        }
    }
    return median;
  }
}
```

Output

2.0

Question 5: Maximum Product Subarray

Copy code

```
public class Main {
    public static void main(String[] args) {
        int[] nums = {2, 3, -2, 4};
        System.out.println(maxProduct(nums));
    }

    public static int maxProduct(int[] nums) {
        if (nums.length == 0) return 0;
        int maxSoFar = nums[0], minSoFar = nums[0], result = nums[0];
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] < 0) {
                int temp = maxSoFar;
                maxSoFar = minSoFar;
                minSoFar = temp;
            }
            maxSoFar = Math.max(nums[i], maxSoFar * nums[i]);
            minSoFar = Math.min(nums[i], minSoFar * nums[i]);
            result = Math.max(result, maxSoFar);
        }
        return result;
    }
}
```

Output

6

Level 4 (Expert)

Problem Statement: Question 1

Given a list of buildings where each building is represented by triplets [Li,Ri,Hi] :

- Li: An integer, representing the left position of the building.
- Ri: An integer, representing the right position of the building.
- Hi: An integer, representing the height of the building.

Calculate the skyline formed by these buildings.

Input Format:

- A list of n building triplets. Each triplet contains three integers.
- $1 \leq n \leq 10^5$
- $0 \leq Li, Ri, Hi \leq 10^4$

Output Format:

- A list of key-height pairs representing the skyline.

Solution:

Copy code

```
import java.util.*;
import java.util.stream.*;

public class Main {

    public static void main(String[] args) {
        int[][] buildings = {{2, 9, 10}, {3, 7, 15}, {5, 12, 12}, {15, 20, 10}, {19, 24, 8}};
        List<List<Integer>> result = getSkyline(buildings);
        for (List<Integer> point : result) {
            System.out.println(point);
        }
    }
```

```
public static List<List<Integer>> getSkyline(int[][] buildings) {
    List<int[]> heights = new ArrayList<>();
    for (int[] b : buildings) {
        heights.add(new int[]{b[0], -b[2]}); // Start of a building
        heights.add(new int[]{b[1], b[2]}); // End of a building
    }

    Collections.sort(heights, (a, b) -> {
        if (a[0] != b[0]) return a[0] - b[0];
        return a[1] - b[1];
    });

    PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> (b - a));
    pq.offer(0);
    int prev = 0;
    List<List<Integer>> result = new ArrayList<>();

    for (int[] h : heights) {
        if (h[1] < 0) {
            pq.offer(-h[1]); // Height of a building start
        } else {
            pq.remove(h[1]); // Height of a building end
        }
        int cur = pq.peek();
        if (prev != cur) {
            result.add(Arrays.asList(h[0], cur)); // A point in the skyline
            prev = cur;
        }
    }

    return result;
}
}
```

```
}
```

Output

[2, 10]
[3, 15]
[7, 12]
[12, 0]
[15, 10]
[20, 8]
[24, 0]

Problem Statement: Question 2

Given a 2D matrix of dimensions m x n containing integers, find the rectangle (sub-matrix) with the maximum sum of its elements.

Input Format:

- A 2D matrix of size m x n where $1 \leq m, n \leq 100$
- Matrix values: $-10^4 \leq matrix[i][j] \leq 10^4$

Output Format:

- An integer representing the maximum sum.

Solution:

Copy code

```
import java.util.Arrays;
import java.util.TreeSet;

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

```
        int[][] matrix = {
            {1, 2, -1, -4, -20},
            {-8, -3, 4, 2, 1},
            {3, 8, 10, 1, 3},
            {-4, -1, 1, 7, -6}
        };
        System.out.println("Maximum Sum of Sub-matrix: " + maxSumRectangle(matrix));
    }

    public static int maxSumRectangle(int[][] M) {
        int m = M.length, n = M[0].length;
        int[] sums = new int[m];
        int maxSum = Integer.MIN_VALUE;

        for (int left = 0; left < n; left++) {
            Arrays.fill(sums, 0);

            for (int right = left; right < n; right++) {
                for (int i = 0; i < m; i++) {
                    sums[i] += M[i][right];
                }

                TreeSet<Integer> set = new TreeSet<>();
                set.add(0);
                int curSum = 0;

                for (int sum : sums) {
                    curSum += sum;
                    Integer num = set.ceiling(curSum - maxSum);
                    if (num != null) {
                        maxSum = Math.max(maxSum, curSum - num);
                    }
                    set.add(curSum);
                }
```

```
            }
        }
    }
    return maxSum;
  }
}
```

Output

Maximum Sum of Sub-matrix: 1

Nested If Else in Java | About, Syntax, Flowchart and... Examples

Have you ever wondered how programs make decisions and adapt to different situations? The answer lies in the power of Nested If else statements. These statements allow us to control...**read more**

Java Comments | About, Types and Examples

Do you know what makes the code more readable? Comments, as they provide valuable context and explanations about the code, making it easier for both the original developers and others...**read more**

Star Pattern Programs in Java

Pattern programs in Java are a type of problem that uses nested loops to produce different patterns of numbers, stars (*), or other characters. In this blog, we will dive...**read more**

Number Pattern Programs in Java

Pattern programs in Java are a type of problem that use nested loops to produce different patterns of numbers, stars (*), or other characters. In this blog we will dive...**read more**

A Guide to Power Function in Java

Have you ever wondered how mathematical power functions are implemented in programming languages like Java? In Java, the Math.pow() function is a powerful tool used to raise a number to...**read more**

Thus, I hope all the questions help you understand the concept better. Keep learning, Keep exploring!

FAQs

What is an array in Java and how do you declare one?

How do you initialize an array in Java?

What is a multidimensional array and how is it used in Java?

How do you iterate over an array in Java?

How do you handle exceptions when working with arrays in Java?