**Arrays**

In Java, an array is a data structure that allows you to store a fixed-size collection of elements of the same data type.

Arrays are commonly used to store and manipulate data in a structured way. Here are some key concepts related to arrays in Java:

1. **Declaration:** To declare an array in Java, you specify the data type of the elements it will hold, followed by the array's name and square brackets **[]**. For example:

int[] myArray; // Declares an integer array named myArray

1. **Initialization**: After declaring an array, you can initialize it by specifying its size and creating an instance of the array using the **new** keyword. For example:

myArray = new int[5]; // Initializes an integer array of size 5

Alternatively, you can declare and initialize an array in a single line:

int[] myArray = new int[5];

1. **Accessing Elements:** Elements in an array are accessed using their index, starting from 0. For example, to access the first element of **myArray**, you would use **myArray[0]**. Here's an example:

int firstElement = myArray[0]; // Accesses the first element

1. **Length:** You can obtain the length (i.e., the number of elements) of an array using the **length** property:

int arrayLength = myArray.length; // Gets the length of myArray

1. **Iterating Over an Array:** You can use loops (e.g., **for** or **foreach**) to iterate over the elements of an array and perform operations on them. For example:

for (int i = 0; i < myArray.length; i++) { // Access and process each element using myArray[i] }

1. **Array Initialization with Values:** You can initialize an array with specific values at the time of declaration using an array initializer:

int[] myArray = {1, 2, 3, 4, 5}; // Initializes an array with specific values

1. **Multidimensional Arrays:** Java supports multidimensional arrays, which are arrays of arrays. You can create 2D, 3D, or even higher-dimensional arrays. For example:

int[][] twoDimArray = new int[3][4]; // Creates a 3x4 2D array

1. **Arrays are Fixed Size:** In Java, once an array is created, its size is fixed and cannot be changed. If you need a dynamic collection of elements, you may consider using other data structures like ArrayList or LinkedList.
2. Array Index Out of Bounds: Be cautious when accessing elements in an array to avoid going out of bounds. Accessing an element outside the valid index range will result in an **ArrayIndexOutOfBoundsException**.
3. Array Types: You can create arrays of any data type in Java, including primitive types (e.g., int, double) and reference types (e.g., objects or other arrays).

**Examples**

**1. Basic Array Initialization and Traversal**

public class BasicArrayExample {

public static void main(String[] args) {

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

System.out.println("Array elements:");

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

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

}

}

}

* **Explanation:** This program initializes an integer array and prints each element using a for loop.

**2. Sum of Array Elements**

public class SumArray {

public static void main(String[] args) {

int[] numbers = {5, 15, 25, 35, 45};

int sum = 0;

for (int num : numbers) {

sum += num;

}

System.out.println("Sum of array elements: " + sum);

}

}

* **Explanation:** Uses an enhanced for loop to sum all the elements of the array.

**3. Finding the Maximum Element in an Array**

public class MaxElement {

public static void main(String[] args) {

int[] numbers = {12, 45, 67, 23, 89, 34};

int max = numbers[0];

for (int i = 1; i < numbers.length; i++) {

if (numbers[i] > max) {

max = numbers[i];

}

}

System.out.println("Maximum element in the array: " + max);

}

}

* **Explanation:** This program finds the maximum element by comparing each element with the current maximum.

**4. Reversing an Array**

public class ReverseArray {

public static void main(String[] args) {

int[] numbers = {1, 2, 3, 4, 5};

System.out.println("Original Array:");

for (int num : numbers) {

System.out.print(num + " ");

}

System.out.println("\nReversed Array:");

for (int i = numbers.length - 1; i >= 0; i--) {

System.out.print(numbers[i] + " ");

}

}

}

* **Explanation:** Prints the array in reverse order using a backward for loop.

**5. Copying One Array to Another**

public class CopyArray {

public static void main(String[] args) {

int[] original = {100, 200, 300, 400, 500};

int[] copy = new int[original.length];

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

copy[i] = original[i];

}

System.out.println("Copied Array:");

for (int num : copy) {

System.out.print(num + " ");

}

}

}

* **Explanation:** Demonstrates how to copy elements from one array to another using a for loop.

**6. Sorting an Array (Bubble Sort Algorithm)**

public class BubbleSort {

public static void main(String[] args) {

int[] arr = {64, 34, 25, 12, 22, 11, 90};

for (int i = 0; i < arr.length - 1; i++) {

for (int j = 0; j < arr.length - 1 - i; j++) {

if (arr[j] > arr[j + 1]) {

// Swap arr[j] and arr[j + 1]

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

System.out.println("Sorted Array (Bubble Sort):");

for (int num : arr) {

System.out.print(num + " ");

}

}

}

* **Explanation:** Implements the Bubble Sort algorithm to sort an array in ascending order.

**7. Binary Search in a Sorted Array**

import java.util.Scanner;

public class BinarySearch {

public static void main(String[] args) {

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

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number to search: ");

int key = sc.nextInt();

int low = 0, high = arr.length - 1;

boolean found = false;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == key) {

found = true;

System.out.println("Element found at index: " + mid);

break;

} else if (arr[mid] < key) {

low = mid + 1;

} else {

high = mid - 1;

}

}

if (!found) {

System.out.println("Element not found.");

}

}

}

* **Explanation:** Searches for an element in a sorted array using the Binary Search algorithm.

**8. Matrix Addition (2D Arrays)**

public class MatrixAddition {

public static void main(String[] args) {

int[][] A = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int[][] B = {

{9, 8, 7},

{6, 5, 4},

{3, 2, 1}

};

int[][] C = new int[3][3];

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

C[i][j] = A[i][j] + B[i][j];

}

}

System.out.println("Resultant Matrix (Addition):");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

System.out.print(C[i][j] + " ");

}

System.out.println();

}

}

}

* **Explanation:** Adds two 3x3 matrices and displays the result.

**9. Transpose of a Matrix**

public class MatrixTranspose {

public static void main(String[] args) {

int[][] matrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int[][] transpose = new int[3][3];

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

transpose[j][i] = matrix[i][j];

}

}

System.out.println("Original Matrix:");

for (int[] row : matrix) {

for (int val : row) {

System.out.print(val + " ");

}

System.out.println();

}

System.out.println("Transposed Matrix:");

for (int[] row : transpose) {

for (int val : row) {

System.out.print(val + " ");

}

System.out.println();

}

}

}

* **Explanation:** Transposes a 3x3 matrix (i.e., rows become columns and vice versa).

**10. Finding Duplicate Elements in an Array**

public class FindDuplicates {

public static void main(String[] args) {

int[] arr = {4, 5, 6, 7, 4, 6, 8, 9};

System.out.println("Duplicate elements in the array:");

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

for (int j = i + 1; j < arr.length; j++) {

if (arr[i] == arr[j]) {

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

break;

}

}

}

}

}

* **Explanation:** Identifies and prints duplicate elements in an array.

1. **Rotate an Array**

public class RotateArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5, 6, 7};

int d = 3;

int n = arr.length;

d = d % n;

// Reverse the first d elements

int start = 0, end = d - 1;

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

// Reverse the remaining elements

start = d;

end = n - 1;

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

// Reverse the whole array

start = 0;

end = n - 1;

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

// Display rotated array

for (int num : arr) {

System.out.print(num + " ");

}

}

}

1. **Spiral Matrix Traversal**

public class SpiralMatrix {

public static void main(String[] args) {

int[][] matrix = {

{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12},

{13, 14, 15, 16}

};

int top = 0, bottom = matrix.length - 1;

int left = 0, right = matrix[0].length - 1;

while (top <= bottom && left <= right) {

for (int i = left; i <= right; i++) {

System.out.print(matrix[top][i] + " ");

}

top++;

for (int i = top; i <= bottom; i++) {

System.out.print(matrix[i][right] + " ");

}

right--;

if (top <= bottom) {

for (int i = right; i >= left; i--) {

System.out.print(matrix[bottom][i] + " ");

}

bottom--;

}

if (left <= right) {

for (int i = bottom; i >= top; i--) {

System.out.print(matrix[i][left] + " ");

}

left++;

}

}

}

}

1. **Merge Two Sorted Arrays**

public class MergeSortedArrays {

public static void main(String[] args) {

int[] arr1 = {1, 5, 9, 10, 15, 20};

int[] arr2 = {2, 3, 8, 13};

int m = arr1.length;

int n = arr2.length;

for (int i = 0; i < m; i++) {

if (arr1[i] > arr2[0]) {

int temp = arr1[i];

arr1[i] = arr2[0];

arr2[0] = temp;

int first = arr2[0];

int k;

for (k = 1; k < n && arr2[k] < first; k++) {

arr2[k - 1] = arr2[k];

}

arr2[k - 1] = first;

}

}

for (int num : arr1) System.out.print(num + " ");

for (int num : arr2) System.out.print(num + " ");

}

}

1. **Kadane's Algorithm (Maximum Subarray Sum)**

public class MaximumSubarray {

public static void main(String[] args) {

int[] arr = {-2, 1, -3, 4, -1, 2, 1, -5, 4};

int maxSoFar = arr[0];

int maxEndingHere = arr[0];

for (int i = 1; i < arr.length; i++) {

maxEndingHere = Math.max(arr[i], maxEndingHere + arr[i]);

maxSoFar = Math.max(maxSoFar, maxEndingHere);

}

System.out.println("Maximum Subarray Sum: " + maxSoFar);

}

}

1. **Sudoku Validator**

public class SudokuValidator {

public static void main(String[] args) {

int[][] board = {

{5, 3, 0, 0, 7, 0, 0, 0, 0},

{6, 0, 0, 1, 9, 5, 0, 0, 0},

{0, 9, 8, 0, 0, 0, 0, 6, 0},

{8, 0, 0, 0, 6, 0, 0, 0, 3},

{4, 0, 0, 8, 0, 3, 0, 0, 1},

{7, 0, 0, 0, 2, 0, 0, 0, 6},

{0, 6, 0, 0, 0, 0, 2, 8, 0},

{0, 0, 0, 4, 1, 9, 0, 0, 5},

{0, 0, 0, 0, 8, 0, 0, 7, 9}

};

boolean[][] rows = new boolean[9][9];

boolean[][] cols = new boolean[9][9];

boolean[][] boxes = new boolean[9][9];

boolean isValid = true;

for (int i = 0; i < 9; i++) {

for (int j = 0; j < 9; j++) {

int num = board[i][j];

if (num != 0) {

num--;

int boxIndex = (i / 3) \* 3 + (j / 3);

if (rows[i][num] || cols[j][num] || boxes[boxIndex][num]) {

isValid = false;

break;

}

rows[i][num] = cols[j][num] = boxes[boxIndex][num] = true;

}

}

if (!isValid) break;

}

System.out.println("Is the Sudoku valid? " + isValid);

}

}

1. **Find the Missing Number in an Array**

public class MissingNumber {

public static void main(String[] args) {

int[] arr = {1, 2, 4, 6, 3, 7, 8};

int n = 8; // The array should have numbers from 1 to 8

int totalSum = n \* (n + 1) / 2; // Sum of first n natural numbers

int arraySum = 0;

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

arraySum += arr[i];

}

int missingNumber = totalSum - arraySum;

System.out.println("Missing Number: " + missingNumber);

}

}

1. **Find the Duplicate Number**

public class FindDuplicate {

public static void main(String[] args) {

int[] arr = {3, 1, 3, 4, 2};

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

int index = Math.abs(arr[i]) - 1;

if (arr[index] < 0) {

System.out.println("Duplicate Number: " + (index + 1));

break;

}

arr[index] = -arr[index];

}

}

}

1. **Count Inversions in an Array**

*(An inversion is when a pair of elements are out of order)*

public class CountInversions {

public static void main(String[] args) {

int[] arr = {1, 20, 6, 4, 5};

int count = 0;

for (int i = 0; i < arr.length - 1; i++) {

for (int j = i + 1; j < arr.length; j++) {

if (arr[i] > arr[j]) {

count++;

}

}

}

System.out.println("Number of Inversions: " + count);

}

}

1. **Trapping Rainwater Problem**

public class TrappingRainwater {

public static void main(String[] args) {

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

int n = height.length;

int waterTrapped = 0;

int[] leftMax = new int[n];

int[] rightMax = new int[n];

leftMax[0] = height[0];

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

leftMax[i] = Math.max(leftMax[i - 1], height[i]);

}

rightMax[n - 1] = height[n - 1];

for (int i = n - 2; i >= 0; i--) {

rightMax[i] = Math.max(rightMax[i + 1], height[i]);

}

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

waterTrapped += Math.min(leftMax[i], rightMax[i]) - height[i];

}

System.out.println("Trapped Rainwater: " + waterTrapped);

}

}

1. **Longest Consecutive Sequence**

import java.util.Arrays;

public class LongestConsecutiveSequence {

public static void main(String[] args) {

int[] arr = {100, 4, 200, 1, 3, 2};

Arrays.sort(arr);

int longestStreak = 1;

int currentStreak = 1;

for (int i = 1; i < arr.length; i++) {

if (arr[i] != arr[i - 1]) { // Ignore duplicates

if (arr[i] == arr[i - 1] + 1) {

currentStreak++;

} else {

longestStreak = Math.max(longestStreak, currentStreak);

currentStreak = 1;

}

}

}

longestStreak = Math.max(longestStreak, currentStreak);

System.out.println("Longest Consecutive Sequence Length: " + longestStreak);

}

}

1. **Product of Array Except Self**

public class ProductExceptSelf {

public static void main(String[] args) {

int[] nums = {1, 2, 3, 4};

int n = nums.length;

int[] left = new int[n];

int[] right = new int[n];

int[] output = new int[n];

left[0] = 1;

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

left[i] = nums[i - 1] \* left[i - 1];

}

right[n - 1] = 1;

for (int i = n - 2; i >= 0; i--) {

right[i] = nums[i + 1] \* right[i + 1];

}

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

output[i] = left[i] \* right[i];

}

for (int num : output) {

System.out.print(num + " ");

}

}

}

1. **Sort an Array of 0s, 1s, and 2s (Dutch National Flag Problem)**

public class Sort012 {

public static void main(String[] args) {

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

int low = 0, mid = 0, high = arr.length - 1;

while (mid <= high) {

if (arr[mid] == 0) {

int temp = arr[low];

arr[low] = arr[mid];

arr[mid] = temp;

low++;

mid++;

} else if (arr[mid] == 1) {

mid++;

} else {

int temp = arr[mid];

arr[mid] = arr[high];

arr[high] = temp;

high--;

}

}

for (int num : arr) {

System.out.print(num + " ");

}

}

}