**Unit-4 Notes**

**Arrays in Java**

An array is a collection of variables of the same type stored in contiguous memory locations. In Java, arrays are objects that can hold a fixed number of elements of a specific type.

**1. Declaring and Initializing Arrays**

**Declaration:** You can declare an array by specifying the type of its elements followed by square brackets [].

**Syntax:**

type[] arrayName;

**Example:**

int[] numbers;

**Initialization:** You can initialize an array using the new keyword.

**Syntax:**

arrayName = new type[size];

**Example:**

numbers = new int[5];

You can also declare and initialize an array in a single line.

**Example:**

int[] numbers = new int[5];

**2. Array Values**

Array elements are accessed by their index, which starts from 0.

**Example:**

int[] numbers = new int[5];

numbers[0] = 10;

numbers[1] = 20;

numbers[2] = 30;

numbers[3] = 40;

numbers[4] = 50;

System.out.println(numbers[0]); // Output: 10

System.out.println(numbers[1]); // Output: 20

**3. Memory Storage Structure**

When an array is created, a block of memory is allocated to hold the array elements. Each element of the array is stored in a contiguous memory location. The base address of the array is the address of the first element.

**Example:**

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

// Memory addresses might look like this (hypothetical):

// numbers[0]: 0x1000

// numbers[1]: 0x1004

// numbers[2]: 0x1008

// numbers[3]: 0x100C

// numbers[4]: 0x1010

**4. Types of Arrays**

**a. Single-Dimensional Arrays**

A single-dimensional array is a list of elements of the same type.

**Example:**

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

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

System.out.println("Element at index " + i + ": " + numbers[i]);

}

**b. Multi-Dimensional Arrays**

A multi-dimensional array is an array of arrays. The most common type is the two-dimensional array, which can be thought of as a table with rows and columns.

**Declaration and Initialization:**

int[][] matrix = new int[3][3];

**Example:**

int[][] matrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

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

for (int j = 0; j < matrix[i].length; j++) {

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

}

System.out.println();

}

**c. Jagged Arrays**

A jagged array is an array of arrays where the sub-arrays can have different lengths.

**Declaration and Initialization:**

int[][] jaggedArray = new int[3][];

jaggedArray[0] = new int[2];

jaggedArray[1] = new int[3];

jaggedArray[2] = new int[4];

**Example:**

int[][] jaggedArray = {

{1, 2},

{3, 4, 5},

{6, 7, 8, 9}

};

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

for (int j = 0; j < jaggedArray[i].length; j++) {

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

}

System.out.println();

}

**Array Length**

You can get the length of an array using the .length attribute.

**Example:**

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

System.out.println("Length of the array: " + numbers.length);

**Common Operations on Arrays**

**a. Traversing an Array**

**Example:**

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

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

System.out.println("Element at index " + i + ": " + numbers[i]);

}

**b. Summing Elements of an Array**

**Example:**

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

int sum = 0;

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

sum += numbers[i];

}

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

**c. Finding the Maximum Element**

**Example:**

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

int max = numbers[0];

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

if (numbers[i] > max) {

max = numbers[i];

}

}

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

**Passing Arrays to Methods**

You can pass arrays to methods as arguments.

**Example:**

public class Main {

public static void main(String[] args) {

int[] numbers = {10, 20,

30, 40, 50}; printArray(numbers); }

public static void printArray(int[] array) {

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

System.out.println("Element at index " + i + ": " + array[i]);

}

}

}

### Returning Arrays from Methods

A method can also return an array.

public class Main {

public static void main(String[] args) {

int[] numbers = getArray();

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

System.out.println("Element at index " + i + ": " + numbers[i]);

}

}

public static int[] getArray() {

return new int[]{10, 20, 30, 40, 50};

}

}

**Cloning Arrays**

You can create a copy of an array using the clone() method.

**Example:**

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

int[] clonedNumbers = numbers.clone();

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

System.out.println("Element at index " + i + ": " + clonedNumbers[i]);

}

**Array Class**

Java provides the java.util.Arrays class that contains various static methods for working with arrays, such as sorting and searching.

**Sorting an Array:**

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

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

Arrays.sort(numbers);

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

System.out.println("Element at index " + i + ": " + numbers[i]);

}

}

}

**Searching an Array:**

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

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

int index = Arrays.binarySearch(numbers, 30);

System.out.println("Index of element 30: " + index);

}

}

**Summary**

* **Arrays**: A collection of variables of the same type stored in contiguous memory locations.
* **Single-dimensional arrays**: Simple list of elements.
* **Multi-dimensional arrays**: Arrays of arrays, e.g., two-dimensional arrays (tables).
* **Jagged arrays**: Arrays of arrays where sub-arrays can have different lengths.
* **Array operations**: Traversing, summing, finding max, passing to methods, cloning, etc.
* **Array class**: Utility methods for sorting, searching, etc.