**Java Arrays**

To declare an array, define the variable type with **square brackets**:

String[] cars;

String[] cars= new String[5]; //String array declaration with size

We have now declared a variable that holds an array of strings. To insert values to it, you can place the values in a comma-separated list, inside curly braces:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"}; This is hard-coded

To create an array of integers, you could write:

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

**Both at same Declare and intilize**

String[] cars = new String[]{"Volvo", "BMW", "Ford", "Mazda"};

*// declare the variable and allocate memory*

int[] myArray = new int[4];

We can edit array elements.

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

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

how to access

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars[0]);

// Outputs Volvo

We can assign other value to array elements.

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

cars[0] = "Opel";

System.out.println(cars[0]);

// Now outputs Opel instead of Volvo

Array Methods

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

System.out.println(cars.length);

// Outputs 4

# **Java Multi-Dimensional Arrays**

**Declare and insitilize at same time in 2D array.**

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

**Loop through 2D Array.**

public class Main {

public static void main(String[] args) {

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

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

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

System.out.println(myNumbers[i][j]);

}  
 }

}

}

The **Arrays** class in **java.util package** is a part of the **Java Collection Framework**. This class provides static methods to dynamically create and access **Java arrays**. It consists of only static methods and the methods of Object class. The methods of this class can be used by the class name itself.

The class hierarchy is as follows:

java.lang.Object

? java.util.Arrays

**Array Sort**

int[] array = {3, 2, 1};

Arrays.sort(array);

for (int i : array) {

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

}

### **Arrays equals**

int[] array1 = {1, 2, 3};

int[] array2 = {1, 2, 3};

boolean isEqual = Arrays.equals(array1, array2);

System.out.println(isEqual);

### **Arrays binarySearch**

but requires the array to be sorted first.

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

int index = Arrays.binarySearch(array, 3);

System.out.println(index);

## Exploring Alternative Approaches to Array Manipulation

### Using ArrayLists for Dynamic Arrays

While Java arrays are powerful, they have a limitation: their size is fixed at the time of creation. To overcome this, we can use ArrayList, a resizable array implementation in the Java Collections Framework.