# 📚 Java Learning Journey

Chapter 8 - Multidimensional Arrays

---

✍️ **Prepared by:** Ahmed Elsifi

## 📌 Introduction

- **Two-dimensional arrays** represent tabular/matrix data (e.g., distance tables, Sudoku grids).
- Example:

  ```
  double[][] distances = {
      {0, 983, 787, 714, 1375, 967, 1087},
      {983, 0, 214, 1102, 1763, 1723, 1842},
      // ... more rows
  };
  ```

---

## 🧩 Declaring Multidimensional Arrays

### 2D Arrays

```
// Preferred syntax
elementType[][] arrayName;

// Alternative (less common)
elementType arrayName[][];
```

**Example:**

```
int[][] matrix;          // Declaration
matrix = new int[5][5]; // Creation
```

### 3D+ Arrays

```
// 3D array
elementType[][][] arrayName = new elementType[x][y][z];
```

**Example:**

```
double[][][] scores = new double[6][5][2]; // 6 students, 5 exams, 2 parts
```

## 🔢 Key Concepts

### 1. Accessing Elements

- Use row and column indices: `array[row][column]`
- **Note:** Java uses separate brackets (not `matrix[2,1]`).

### 2. Array Lengths

- `array.length` → number of rows.
- `array[i].length` → number of columns in row `i`.

### 3. Ragged Arrays

- Rows can have different lengths:

```java
int[][] ragged = {
    {1, 2, 3},
    {4, 5},
    {6}
};
```

---

## 🔄 Processing 2D Arrays

Use nested loops for:

- **Initialization** (input/random values).
- **Printing** elements.
- **Summing** all elements or by row/column.
- **Finding max/min** values.
- **Shuffling** elements.

---

## 📤 Passing to Methods

- Pass by reference (like 1D arrays).
- Example method:

```java
public static int sum(int[][] m) {
    int total = 0;
    for (int[] row : m)
        for (int val : row)
            total += val;
    return total;
}
```

## 🌐 Multidimensional Arrays

- A 3D array is an array of 2D arrays.
- Example: Weather data (days × hours × temperature/humidity).

---

## 💡 Important Notes

- Use `array[i][j]` to access elements.
- Rows can have different lengths (ragged arrays).
- Nested loops are essential for processing.
- Arrays are passed by reference to methods.
- Multidimensional arrays generalize to n dimensions.

---

## ❗ Common Mistakes

- Using `matrix[2,1]` instead of `matrix[2][1]`.
- Forgetting that rows can have different lengths.
- Not using nested loops for full traversal.