

Java 2-Dimensional Arrays

What is a 2D Array?

A **2-dimensional array** (or 2D array) is an array of arrays. It organizes data in a grid with rows and columns, just like a table or spreadsheet. While a 1D array is a single line of elements, a 2D array has both width and height, making it perfect for storing data that naturally fits in a grid structure.

Simple Analogy: 2D Arrays are Like a Spreadsheet or Chess Board

Think of a 2D array like different grid-based items you already know:

- **Spreadsheet** = Rows and columns of data (like Excel)
- ♔ **Chess Board** = 8 rows × 8 columns of squares
- **Movie Theater Seating** = Rows of seats, each seat has a row and seat number
- 📅 **Calendar** = Weeks (rows) × Days (columns)
- **Multiplication Table** = Numbers arranged in rows and columns

Example: 2D Array "grid" with 3 rows and 4 columns

	col 0	col 1	col 2	col 3
row 0	1	2	3	4
row 1	5	6	7	8
row 2	9	10	11	12

Access element: `grid[1][2]` gets the value **7** (row 1, column 2)

🔑 Key 2D Array Concepts

- **Row:** The horizontal lines (first index)
- **Column:** The vertical lines (second index)
- **Element Access:** `array[row][column]` - row first, then column
- **Dimensions:** rows × columns (e.g., 3×4 array has 3 rows and 4 columns)
- **All rows can have different lengths** (called "jagged arrays")

How to Create 2D Arrays

Method 1: Declare and Create Separately

```
// Step 1: Declare the 2D array
int[][] matrix;

// Step 2: Create with size (3 rows, 4 columns)
```

```
matrix = new int[3][4];
```

```
// Step 3: Assign values
```

```
matrix[0][0] = 1;
```

```
matrix[0][1] = 2;
```

```
matrix[1][0] = 5;
```

```
// etc...
```

Method 2: Declare and Create Together

```
// Create empty 2D array (all values start at 0)
```

```
int[][] scores = new int[5][3]; // 5 rows, 3 columns
```

```
// Different data types
```

```
String[][] names = new String[4][2];
```

```
double[][] prices = new double[3][3];
```

Method 3: Initialize with Values (Most Common!)

```
// Create and fill in one statement
```

```
int[][] grid = {
```

```
    {1, 2, 3, 4}, // row 0
```

```
    {5, 6, 7, 8}, // row 1
```

```
    {9, 10, 11, 12} // row 2
```

```
};
```

```
// Each row is a 1D array!
```

```
// This creates a 3x4 array (3 rows, 4 columns)
```

Method 4: Jagged Arrays (Different Column Lengths)

```
// Each row can have different number of columns
```

```
int[][] jagged = {
```

```
    {1, 2}, // row 0 has 2 elements
```

```
    {3, 4, 5, 6}, // row 1 has 4 elements
```

```
    {7, 8, 9} // row 2 has 3 elements
```

```
};
```

Accessing 2D Array Elements

Reading and Writing 2D Array Values

```
int[][] numbers = {
```

```

    {10, 20, 30},
    {40, 50, 60},
    {70, 80, 90}
};

// Read values - [row][column]
int value1 = numbers[0][0]; // 10 (first row, first column)
int value2 = numbers[1][2]; // 60 (second row, third column)
int value3 = numbers[2][1]; // 80 (third row, second column)

System.out.println("Value at [0][0]: " + value1);
System.out.println("Value at [1][2]: " + value2);
System.out.println("Value at [2][1]: " + value3);

// Write values (modify array)
numbers[0][1] = 99; // Change 20 to 99
numbers[2][2] = 100; // Change 90 to 100

```

2D Array Dimensions and Length

Getting Rows and Columns

```

int[][] table = {
    {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}
};

// Get number of rows
int numRows = table.length; // 3 rows

// Get number of columns in a specific row
int numCols = table[0].length; // 4 columns (in row 0)

System.out.println("Rows: " + numRows);
System.out.println("Columns: " + numCols);
System.out.println("Total elements: " + (numRows * numCols));

// For jagged arrays, check each row separately
int[][] jagged = {{1, 2}, {3, 4, 5}, {6}};
System.out.println("Row 0 length: " + jagged[0].length); // 2
System.out.println("Row 1 length: " + jagged[1].length); // 3
System.out.println("Row 2 length: " + jagged[2].length); // 1

```

Looping Through 2D Arrays

Method 1: Nested For Loops (Most Common)

```

int[][] matrix = {

```

```

    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

// Outer loop: rows
for (int row = 0; row < matrix.length; row++) {
    // Inner loop: columns
    for (int col = 0; col < matrix[row].length; col++) {
        System.out.print(matrix[row][col] + " ");
    }
    System.out.println(); // New line after each row
}

```

Output:

```

1 2 3
4 5 6
7 8 9

```

Method 2: Enhanced For Loop (for-each)

```

String[][] names = {
    {"Alice", "Bob"},
    {"Charlie", "Diana"},
    {"Eve", "Frank"}
};

// Outer loop: each row
for (String[] row : names) {
    // Inner loop: each element in row
    for (String name : row) {
        System.out.print(name + " ");
    }
    System.out.println();
}

```

Note: Enhanced for loop is easier but you don't have access to row/column indexes.

⚠ Common 2D Array Mistakes

Mistake 1: Wrong Index Order

```

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

// WRONG: column first, then row
int value = arr[2][1]; // ArrayIndexOutOfBoundsException!

// CORRECT: row first, then column
int value = arr[1][2]; // Gets 6

```

Mistake 2: Assuming All Rows Have Same Length

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

// WRONG: Assumes all rows have 4 columns
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 4; j++) { // Crashes on row 0!
        System.out.print(jagged[i][j]);
    }
}

// CORRECT: Check each row's length
for (int i = 0; i < jagged.length; i++) {
    for (int j = 0; j < jagged[i].length; j++) { // Safe!
        System.out.print(jagged[i][j]);
    }
}
```

Practical 2D Array Examples

Example 1: Grade Book System

```
import java.util.Scanner;

public class GradeBook {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // 4 students, 3 test scores each
        double[][] grades = new double[4][3];
        String[] studentNames = {"Alice", "Bob", "Charlie", "Diana"};

        // Input grades
        for (int student = 0; student < grades.length; student++) {
            System.out.println("Enter grades for " + studentNames[student]);
            for (int test = 0; test < grades[student].length; test++) {
                sum += grades[student][test];
            }
            double average = sum / grades[student].length;
            System.out.println(studentNames[student] + ": Average = " + average);
        }

        input.close();
    }
}
```

Example 2: Tic-Tac-Toe Board

```
public class TicTacToe {
```

```

public static void main(String[] args) {
    // Create 3x3 board
    char[][] board = {
        {'X', 'O', 'X'},
        {'O', 'X', 'O'},
        {'X', ' ', 'O'}
    };

    // Display board
    System.out.println("Tic-Tac-Toe Board:");
    for (int row = 0; row < board.length; row++) {
        for (int col = 0; col < board[row].length; col++) {
            System.out.print(board[row][col]);
            if (col < board[row].length - 1) {
                System.out.print(" | ");
            }
        }
        System.out.println();
        if (row < board.length - 1) {
            System.out.println("-----");
        }
    }
}

```

Output:

```

Tic-Tac-Toe Board:
X | O | X
-----
O | X | O
-----
X |  | O

```

Example 3: Finding Maximum and Minimum Values

```

int[][] temperatures = {
    {72, 75, 78, 82}, // Week 1
    {68, 71, 74, 76}, // Week 2
    {70, 73, 77, 80} // Week 3
};

// Find highest and lowest temperatures
int max = temperatures[0][0];
int min = temperatures[0][0];
int maxRow = 0, maxCol = 0;
int minRow = 0, minCol = 0;

for (int row = 0; row < temperatures.length; row++) {
    for (int col = 0; col < temperatures[row].length; col++) {
        if (temperatures[row][col] > max) {
            max = temperatures[row][col];
            maxRow = row;
            maxCol = col;
        }
    }
}

```

```

    if (temperatures[row][col] < min) {
        min = temperatures[row][col];
        minRow = row;
        minCol = col;
    }
}

System.out.println("Highest: " + max + "°F (Week " + (maxRow + 1) + ", Day " + (maxCol + 1) +
    ")");
System.out.println("Lowest: " + min + "°F (Week " + (minRow + 1) + ", Day " + (minCol + 1) +
    ")");

```

Example 4: Matrix Addition

```

int[][] matrix1 = {
    {1, 2, 3},
    {4, 5, 6}
};

int[][] matrix2 = {
    {7, 8, 9},
    {10, 11, 12}
};

// Create result matrix
int[][] sum = new int[2][3];

// Add corresponding elements
for (int row = 0; row < matrix1.length; row++) {
    for (int col = 0; col < matrix1[row].length; col++) {
        sum[row][col] = matrix1[row][col] + matrix2[row][col];
    }
}

// Display result
System.out.println("Matrix Sum:");
for (int row = 0; row < sum.length; row++) {
    for (int col = 0; col < sum[row].length; col++) {
        System.out.print(sum[row][col] + " ");
    }
    System.out.println();
}

```

Output:

```

Matrix Sum:
8 10 12
14 16 18

```

Example 5: Seating Chart with Search

```
String[][] classroom = {
    {"Alice", "Bob", "Charlie"},
    {"Diana", "Eve", "Frank"},
    {"Grace", "Henry", "Iris"}
};

// Display seating chart
System.out.println("=== Classroom Seating Chart ===");
for (int row = 0; row < classroom.length; row++) {
    System.out.print("Row " + (row + 1) + ": ");
    for (int col = 0; col < classroom[row].length; col++) {
        System.out.print(classroom[row][col] + " ");
    }
    System.out.println();
}

// Find a student's location
String target = "Eve";
boolean found = false;

for (int row = 0; row < classroom.length && !found; row++) {
    for (int col = 0; col < classroom[row].length; col++) {
        if (classroom[row][col].equals(target)) {
            System.out.println("\n" + target + " is seated in Row " + (row + 1) + ", Seat " + (col + 1));
            found = true;
            break;
        }
    }
}

if (!found) {
    System.out.println("\n" + target + " not found in classroom.");
}
```

Common 2D Array Operations Summary

Operation	Code Pattern	Description
Create rectangular array	int[][] arr = new int[3][4];	3 rows, 4 columns, all values 0
Create with values	int[][] arr = {{1, 2}, {3, 4}};	Initialize with specific values
Get number of rows	int rows = arr.length;	Returns total number of rows
Get number of columns	int cols = arr[0].length;	Returns columns in first row
Access element	int val = arr[row][col];	Gets value at [row][col]
Modify element	arr[row][col] = 99;	Sets value at [row][col]
Loop through all	Nested for loops	Outer loop: rows, Inner loop: columns

2D Array Best Practices

- **Always use `arr.length`** for number of rows
- **Use `arr[row].length`** for columns in that row
- **Remember `[row][column]`** order (not column first!)
- **Check bounds** before accessing elements
- **Use meaningful variable names** (row, col not i, j when clarity helps)
- **Comment complex nested loops**
- **Consider jagged arrays** when rows have different lengths

Common 2D Array Mistakes

- **Wrong index order** (`arr[col][row]` instead of `arr[row][col]`)
- **Off-by-one errors** with array bounds
- **Assuming rectangular arrays** (all rows same length)
- **Confusing `arr.length`** with `arr[0].length`
- **Not checking for empty arrays**
- **Hard-coding array dimensions** instead of using `.length`
- **Forgetting arrays are zero-indexed**

When to Use 2D Arrays

Perfect Uses for 2D Arrays:

- **Game boards** (chess, checkers, tic-tac-toe, battleship)
- **Seating arrangements** (theaters, classrooms, airplanes)
- **Spreadsheet-like data** (grades, sales data, schedules)
- **Image processing** (pixels in rows and columns)
- **Maps and grids** (coordinate systems, mazes)
- **Tables and matrices** (mathematical operations)
- **Monthly calendars** (weeks × days)

Real-World Application: Sales Report

```
public class SalesReport {
    public static void main(String[] args) {
        // Sales data: [stores][months]
        double[][] sales = {
            {5000, 5500, 6000, 6200}, // Store 1
            {4800, 5200, 5400, 5600}, // Store 2
            {6200, 6500, 6800, 7000} // Store 3
        };

        String[] months = {"Jan", "Feb", "Mar", "Apr"};

        // Print header
        System.out.print("Store");
        for (String month : months) {
            System.out.print(month);
        }
        System.out.println("Total");
        System.out.println("-----");
    }
}
```

```
// Print sales data and calculate totals
double grandTotal = 0;

for (int store = 0; store < sales.length; store++) {
    System.out.print("Store " + (store + 1));
    double storeTotal = 0;

    for (int month = 0; month < sales[store].length; month++) {
        System.out.print(sales[store][month]);
        storeTotal += sales[store][month];
    }

    System.out.println(storeTotal);
    grandTotal += storeTotal;
}

System.out.println("-----");
System.out.println("Grand Total: $" + grandTotal);
}
```

Your Turn: Write Your Own Definition

What is a 2-dimensional array in Java? How would you explain it to a friend?

Write your definition in your own words:

Fill in the code to complete these 2D array operations:

```
// Create a 2D array with 4 rows and 3 columns
int[][] grid = _____;

// Set the element at row 2, column 1 to 99
_____ = 99;

// Get the number of rows
int numRows = _____;

// Get the number of columns in row 0
int numCols = _____;

// Print element at row 1, column 2
System.out.println(_____);
```

Draw a simple diagram showing how this 2D array is organized:

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

Draw it here with row and column labels:

Describe three real-world situations where you would use a 2D array:

1. _____
2. _____
3. _____

Explain the difference between `arr.length` and `arr[0].length`: