

# AP Lesson 4.1: The Game of Concentration

## AP ACTIVITY 4.1.1

# Two-Dimensional Arrays (AP)

### INTRODUCTION

Imagine playing a game of checkers or chess. The game board you use for both games is the same: a board of squares, eight squares by eight squares. Game boards arranged in a matrix or a grid are used in many types of games. In many programming languages, a grid-like game board can be represented as a new kind of array called a two-dimensional array.

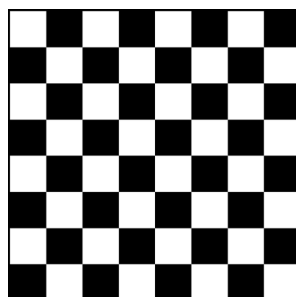
#### Materials

- Computer with BlueJ IDE

## Procedure

### Part I: The Game Board

A **two-dimensional array** (or **2D array**) is a collection of storage locations organized like a matrix, grid, or checkerboard. In the Java® programming language, a 2D array is actually an array of arrays. To help you understand this “array of arrays”, consider each row of a checkerboard, each with eight squares:

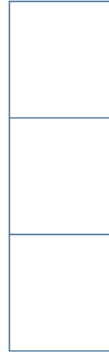


#### 2D Array

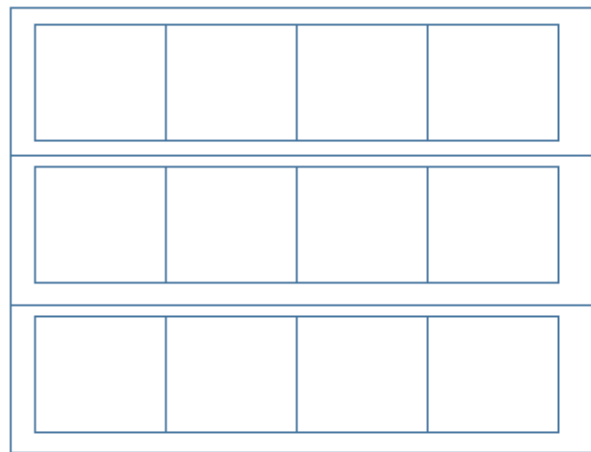
A collection of storage locations organized like a matrix, grid, or checkerboard.

The elements of a 2D array are stored in memory as an outer array where each element contains another array.

For example, a 2D array with three rows and four columns would have an outer array of three elements.



Each outer array element would each contain an inner array of four elements.



This is an “array of arrays” representing a 3 x 4 board or matrix.

## Part II: 2D Arrays in Your Program

---

Declaring a 2D array is similar to declaring a one-dimensional array. You just use another set of brackets to indicate the second dimension.

```
// a 2D array named lastnames that can store Strings
String[][] lastnames = new String[3][4];

// a larger 2D integer array called numbers
int[][] numbers = new int[10][10];

// an even larger array of Objects
Object[][] someObjects = new Object[250][50];
```

To access individual elements of a 2D array in Java, you indicate first the row and then the column. This is called **row-major order**. The row is the more important part since it contains the other array, and it is specified first. You can also remember row-major order by comparing it to how you read a book; you start at the top of the page and read a line (row) at a time, moving your eye left to right over the words (columns).

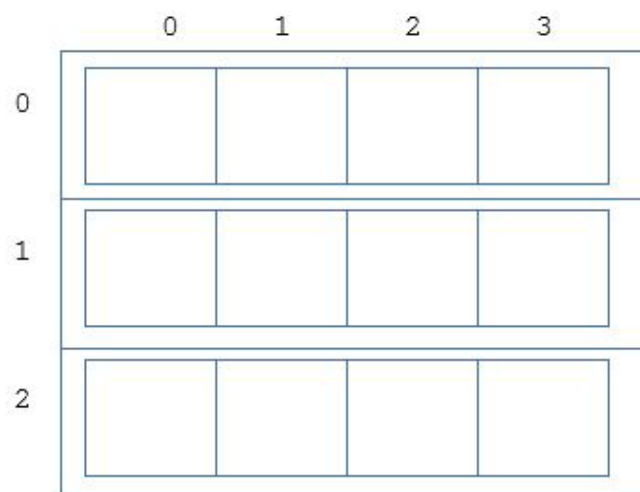
#### row-major ordering

A convention for accessing data in 2-D structures; the first index indicates the row and the second indicates the column.

The following is an example to see this row-column ordering in action. The code to create a board of 12 strings, 3 rows by 4 columns is:

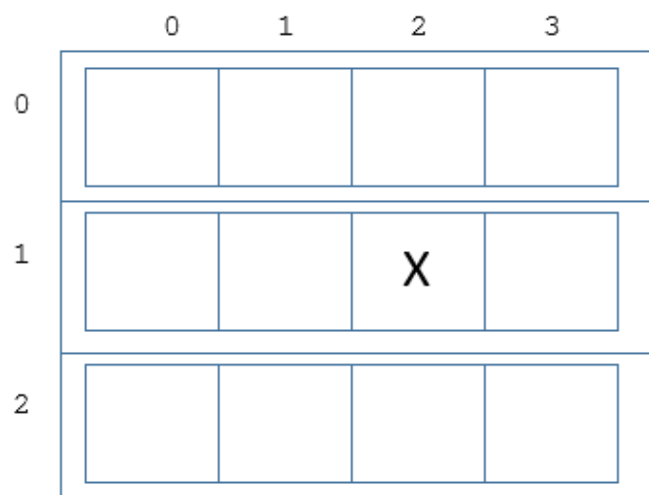
```
String[][] board = new String[3][4];
```

The result in memory is an array of arrays, shown here with their index values:



To reference an individual element in the array, use the name of the array followed by the two index values in brackets. The first element in the top row would be `board[0][0]`. The next element in the top row would be `board[0][1]`.

What is the correct reference for the board element with the X in it?



### Check your answer

```
board[1][2]; // remember row-major order
```

What are the correct references for all of the elements that have an X? Start with the X at the top-most position and parse in RC order.

	0	1	2	3
0		X		
1				
2	X			X

### Check your answer

```
board[0][1];
```

```
board[2][0];
```

```
board[2][3];
```

## Part III: 2D Arrays and Nested Loops

To visit every element of the 2D array, you create a `for` loop to iterate over each row:

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

Then, in each row, you create another `for` loop to iterate over each column:

```
for (int i = 0; i < 3; i++) {  
    for (int j = 0; j < 4; j++) {  
        // each element accessed here, for example print  
        System.out.println( board[i][j] );  
    }  
}
```

This is a very common *nested loop* algorithm. A **nested loop** is a construct that has two loops, one loop inside of another. The nested loops above use the variable `i` to iterate rows and the variable `j` to iterate columns; the individual array element is accessed with `board[i][j]`.

#### nested loop

A looping construct that consists of two loops, one loop inside of another

#### NOTE

Using the variables `i` and `j` in a nested loop is a strong programming convention.

- 1 Write an array algorithm to initialize every element of the `values` array to the empty string.

```
String [][] values = new String[10][25];
```



## Part IV: 2D Array Lengths

---

For 2D arrays, what we might consider length and width are technically two array lengths:

```
rows = board.length; // number of rows - the outer array  
cols = board[0].length; // number of columns - the inner array
```

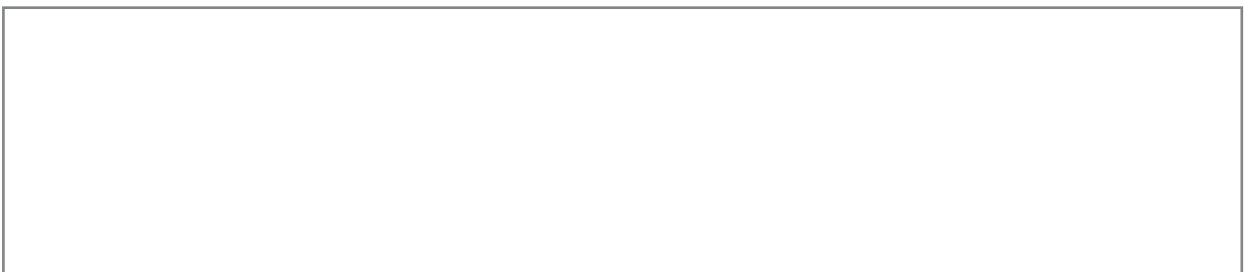
### NOTE

Other rows could also be used to get the length of the inner array (the number of columns). Using row 0 is another strong Java convention.

When you write algorithms for 2D arrays, it is always best to use the array `length` variable instead of hard coding values, such as “3” or “4”.

- 2 Rewrite the algorithm you wrote above without the hard-coded values.

```
String [][] values = new String[10][25];
```

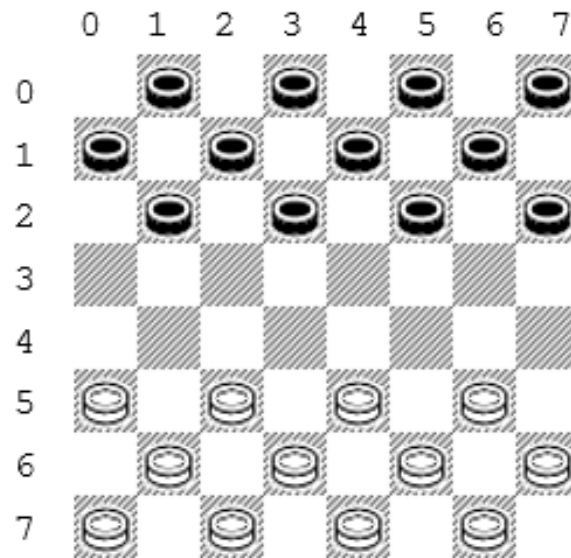


## Part V: A 2D Array Algorithm

---

In this part of the activity, you will write an array algorithm using a 2D `String` array called `checkerBoard`. To represent the state of the board at the start of a checkers game, assign the string values “BLACK” and “WHITE” to the appropriate array elements.

- 3 Review how checkers are placed on the board and the indexes Java uses to reference the board:



- 4 Now think algorithmically how checkers are placed on the board. Consider the indexes of the rows and columns that represent the board, and whether rows and columns are odd or even. Complete the pseudocode below:

***checkerboard is 8 rows by 8 columns  
in first three rows***

***in last three rows***

- 5 Convert the pseudocode to a Java algorithm. Create a new BlueJ Project and **Class**, both called “Checkers”. In the Checkers class, replace the auto-generated BlueJ code with the code below and complete the algorithm.

#### class

A collection of code that serves a common purpose.

```
public class Checkers {
    public static void main( ) {
        String[][] checkerBoard = new String[8][8];

        for (int i = 0; i < checkerBoard.length; i++)
            int oddRow = i % 2;
        for (int j = 0; j < checkerBoard[0].length; j++)
            int oddColumn = j % 2;

            // Assign "BLACK" or "WHITE" to the appropriate squares

        }

        // Print out the board as 8 rows with either null,
        // "BLACK", or "WHITE" in each element
        // Hint: use both the println and print methods
        // of System.out

    }
}
```

## CONCLUSION

1. How did algorithmic thinking help you to write pseudocode?
2. How did writing pseudocode help you write your solution?