

# INFO1103: Introduction to Programming

School of Information Technologies, University of Sydney



## Week 12: Multi-dimensional arrays, More abstractions

**We will cover:** Traversal of multi-dim arrays, Interfaces and Abstract class

**You should read:** §§7.5, 8.4

## Lecture 23: Multi-dimensional arrays

*Arrays of arrays, and more*

# Multiple dimensions

Declaring a multi-dimensional array is easy

Each dimension has another index []

e.g. 2D `int [] [] grid;`

e.g. 3D `int [] [] [] volume;`

e.g. 6D `int [] [] [] [] [] [] hyperplane;`

We will keep it simple and use 2D. The same ideas apply when extending to higher dimensions

## 2 dimensions initialisation

Initialise a 2D array

```
1  int [][] grid = new int [2] [2];
```

creates a new array of  $(2 \times 2 =) 4$  ints

We access them in the same way as for 1-dimensional arrays:

```
1  grid[0][0] = 7;  
2  grid[0][1] = 7;  
3  grid[1][0] = 7;  
4  grid[1][1] = 7;
```

Which we can imagine, it might look like this:

$$\begin{pmatrix} 7 & 7 \\ 7 & 7 \end{pmatrix}$$

## 2 dimensions traversal

We need a procedure to visit each element in a 2D array and print the value

Such a procedure is termed a *traversal*

```
1 // initialise
2 int grid [][] = new int [2][2];
3
4 // our traversal to assign the initial value
5 for (int x = 0; x < 2; x++) {
6     for (int y = 0; y < 2; y++) {
7         grid[ x ] [ y ] = 1;
8     }
9 }
10 // our traversal to display values
11 for (int x = 0; x < 2; x++) {
12     for (int y = 0; y < 2; y++) {
13         System.out.print(grid[x][y] + " ");
14     }
15     System.out.println();
16 }
```

## Two-D Arrays — example program

```
1 public class TwoDArray {
2     /*
3     * Create and display a 2D array
4     */
5     public static void main(String [] args) {
6         int grid [][] = new int[5][10];
7         for (int i = 0; i < 5; i++) {
8             for (int j = 0; j < 10; j++) {
9                 System.out.print(grid[i][j] + " ");
10            }
11            System.out.println();
12        }
13    }
14 }
```

# Running TwoDArray

```
~> javac TwoDArray.java
~> java TwoDArray
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
~>
```

...which is rather dull, but works.



# Printing 2D array contents

How we imagine the memory contents and how we print those contents to screen are **two different ideas**

Here is *somebody's* print out of a 2D array:

```
0 3  
7 0
```

What are the values of 2D array elements `[0][1]` and `[1][0]`?

This depends on which traversal is used and if they are the same for both setting and printing.

# Which traversal

Representation:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Output:

1	2
3	4

```
1 public static void init_trace(int[][] grid, int size) {
2     int count = 1;
3     for (int i = 0; i < size; i++) {
4         for (int j = 0; j < size; j++) {
5             grid[i][j] = count;
6             count++;
7         }
8     }
9 }
10 public static void print(int[][] grid, int size) {
11     for (int i = 0; i < size; i++) {
12         for (int j = 0; j < size; j++) {
13             System.out.print(grid[i][j] + " ");
14         }
15         System.out.println();
16     }
17 }
```

# Which traversal

Representation:

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

Output:

1	2
3	4

```
1 public static void init_trace(int[][] grid, int size) {
2     int count = 1;
3     for (int i = 0; i < size; i++) {
4         for (int j = 0; j < size; j++) {
5             grid[i][j] = count;
6             count++;
7         }
8     }
9 }
10 public static void print(int[][] grid, int size) {
11     for (int i = 0; i < size; i++) {
12         for (int j = 0; j < size; j++) {
13             System.out.print(grid[i][j] + " ");
14         }
15         System.out.println();
16     }
17 }
```

Which makes more sense? i.e. what is the correct order to visit each

Consider this example:

```
1 public class TwoDArrayFilled {
2     /*
3      * Create and display a multiplication table :)
4      */
5     public static void main(String [] args) {
6         int grid [][] = new int[5][10];
7         for (int i = 0; i < 5; i++) {
8             for (int j = 0; j < 10; j++) {
9                 grid[i][j] = i*j;
10            }
11        }
12        for (int i = 0; i < 5; i++) {
13            for (int j = 0; j < 10; j++) {
14                System.out.print(grid[i][j] + " ");
15            }
16            System.out.println();
17        }
18    }
19 }
```

# Running TwoDArrayFilled

```
~> javac TwoDArrayFilled.java
~> java TwoDArrayFilled
0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9
0 2 4 6 8 10 12 14 16 18
0 3 6 9 12 15 18 21 24 27
0 4 8 12 16 20 24 28 32 36
~>
```

# Size of a multi-dimensional array

There's an obvious question that you might be wondering about now: what is the value of `grid.length` in the above?

Let's find out. With these lines inserted into our program...

```
1 System.out.println(grid.length);  
2 System.out.println(grid[2].length);
```

we get

```
~> java TwoDArrayFilled  
0 0 0 0 0 0 0 0 0 0  
0 1 2 3 4 5 6 7 8 9  
0 2 4 6 8 10 12 14 16 18  
0 3 6 9 12 15 18 21 24 27  
0 4 8 12 16 20 24 28 32 36  
5  
10
```

## 2 dimensions non-square

Initialise a 2D array

```
1 int [][] grid = new int [2] [4];
```

creates a new array of  $(2 \times 4 =) 8$  ints

What does it look like?  $2 \times 4$  or  $4 \times 2$ ?

$2 \times 4$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$4 \times 2$

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

# Arrays of arrays

To understand what's going on you need to know an important fact: multidimensional arrays are stored as arrays of arrays.



*An array of  $k$  dimensions is stored as a 1-dimensional array of arrays of  $(k - 1)$  dimensions each.*

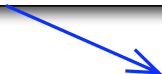
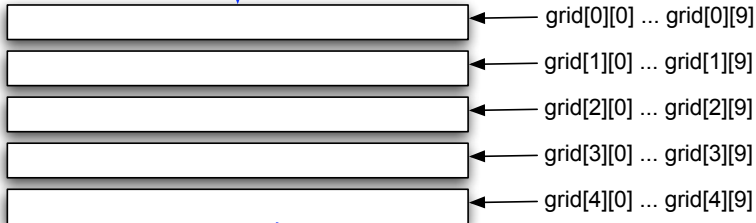
```
int[] [] grid = new int[5][10];  
System.out.println("dim 1:  " + grid.length);  
System.out.println("dim 2:  " + grid[0].length);
```

So in the example above, `grid.length` is the length of the first dimension: it's the number of 1-dimensional arrays

`grid[0][0] ... grid[0][9]` is the first “row”, `grid[1][0] ... grid[1][9]` the second, etc.



5x10 array  
made up of 5 1-Dimensional arrays



# Exception with 2D arrays

```
1 public class BadGrid {  
2     public static void main(String[] args) {  
3         int[][] grid = new int[5][10];  
4  
5         grid[3][2000] = 1; // oops  
6     }  
7 }
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:  
    2000 at BadGrid.main(BadGrid.java:5)
```

There's an *exception thrown* when the JVM tries to access `grid[3][2000]`, as the correct range of indices for `c[3]` is `c[3][0] . . . c[3][9]`.

# Alternative initialisation of array

```
c = new int[5] [];
```

all arrays  $c[i]$  are null!

null
null
null
null
null

```
c[2] = new int[19];
```

$c[2]$  has 19 elements, all 0

```
c[3] = new int[] { 4, 3, 6, 1 };
```

$c[2]$  has 19 elements, all 0

4	3	6	1
---	---	---	---