Arrays

#7

CS 0007
Introduction to
Computer Programming

Luís Oliveira

Summer 2020

Arrays

- Arrays are collections of values OF THE SAME TYPE
  - They are stored consecutively in memory
- To declare an array of `int`s you need to use `new`

```
int theArray[] = new int[10];
```

theArray variable contains the memory address of the start of the array

(1) allocates space in memory to store 10 ints
(2) Returns the memory address to that space in memory

Because the variable holds a memory address, we say it holds a **reference**.

Random starting number

Addr	Value
1350	0
1354	30
1358	04
1362	0
1366	123
1370	20
1374	34
1378	48
1382	78
1386	34
1390	??

The variable name points to memory

```
int[] haHa = new int[10];
```

- To access the value stored in memory
  - We need to index the array
  - Dereference the element address

- This is done with the `[]` operator

Addr	Value
1350	0
1354	30
1358	04
1362	0
1366	123
1370	20
1374	34
1378	48
1382	78
1386	34
1390	??

Access to elements

- Accessing arrays

```
System.out.println(array[0]);
```
→ `0`

```
System.out.println(array[6]);
```
→ `34`

```
System.out.println(array[8]);
```
→ `78`

- Changing values of the array

```
array[0] = 12;
```

- Index 10 will stop the program with an error

```
array[10] = 12;
```

Index	Value
0	~~0~~ 12
1	30
2	04
3	0
4	123
5	20
6	34
7	48
8	78
9	34
10	??

The length of the array

- You can ask the array how big it is

```java
int theArray[] = new int[10];
System.out.println("The array has space for "
                  + theArray.length + "ints.");
```

- This one has space in memory to hold 10 `int`s
  - This space cannot be changed
- Indices start at ZERO!
- So… The last index is not 10! It's 9.

	Index	Value
array[0]	0	0
	1	30
	2	04
	3	0
	4	123
	5	20
array[6]	6	34
	7	48
	8	78
array[9]	9	34
	10	??

Arrays and helpers

- Good practice to create a constant, we don't like magic numbers ☺

```
final int SIZE_OF_ARRAY = 10;
int array[] = new int[SIZE_OF_ARRAY];
```

- You can initialize the array on declaration!

```
int array[] = {0, 30, 4, 0, 123, 20, 34, 48, 78, 34};
```

- If you don't… keep an extra variable with the number of elements
  - Remember the size is fixed, but the number of valid elements may change

```
int numberOfElements = 0;
final int SIZE_OF_ARRAY = 10;
int array[] = new int[SIZE_OF_ARRAY];
```

Keeping track of filled portion

- The variable is useful for and after filling the array

```
int numberOfElements = 0;
final int SIZE_OF_ARRAY = 10;
int array[] = new int[SIZE_OF_ARRAY];

int userInput = getUserInput();
while(userInput>=0) {
    array[numberOfElements] = userInput;
    numberOfElements++;
    userInput = getUserInput();
}
```

Passing arrays to functions

- Functions behave different with arrays
  - They are complex data types (yikes!)

```
public static void changeElement(int[] array, int index) {
    array[index]++;
}
```

```
int[] array = {1,2,3,4,5};
System.out.println(array[4]);
changeElement(array, 4);
System.out.println(array[4]);
```

```
5
```

```
6
```

Instead of making a copy of the array, Java will give the function the array's address in memory!

So changes to the array inside a function will be visible by the caller

2D Arrays

- Representing tables
- Matrices? (Do you know about them?)

```
int[][] 2DArray = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
System.out.println(array[0][1]);
array[0][1] = 2 * array[0][1];
System.out.println(array[0][1]);
```

	Col 0	Col 1	Col 2	Col 3
Row 0	1	2 ➔4	3	4
Row 1	5	6	7	8
Row 2	9	10	11	12

2D Arrays – Arrays of arrays?

- What is?

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

2DArray

	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

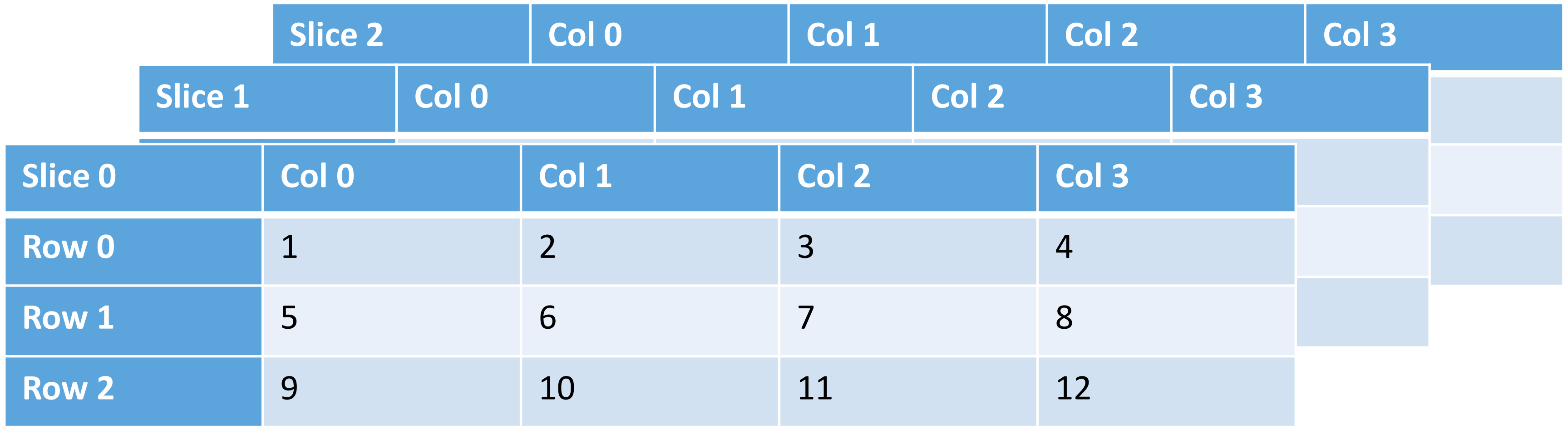
1DArray

Index 0	Index 1	Index 2	Index 3
5	6	7	8

3D???

- Sure… y
  - you can still visualize it as a table. The first dimension is now a slice!

```
int[][][] 3DArray = {{ 1, 2, 3, 4},{ 5, 6, 7, 8},{ 9,10,11,12},
                     {13,14,15,16},{17,18,19,20},{21,22,23,24},
                     {25,26,27,28},{29,30,31,32},{33,34,35,36}};
```



Slice 2	Col 0	Col 1	Col 2	Col 3

Slice 1	Col 0	Col 1	Col 2	Col 3

Slice 0	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

MORE????

- Tracking your expenses?

```
double[][][][] dailyFiveYearExpenses = new double[5][12][31][100];
```

- But it’s now hard to track!
  - This is where the concept of arrays of arrays is useful!

```
double[][][][] dailyFiveYearExpenses = new double[5][12][31][100];
double[][][] y2016 = dailyFiveYearExpenses[0];
double[][] dec2016 = y2016[12-1];//<- :)
double[] christmas2016 = dec2016[25-1];
double firstGift = christmas2016[0];
```