Array and ArrayLists

Comp 401 Recitation 2

Just a Quick Refresher on the Array

Remember arrays?

Type[] array=new Type[size];

int[] array=new int[3];

String[] array=new String[]{“Hello”,“World”};

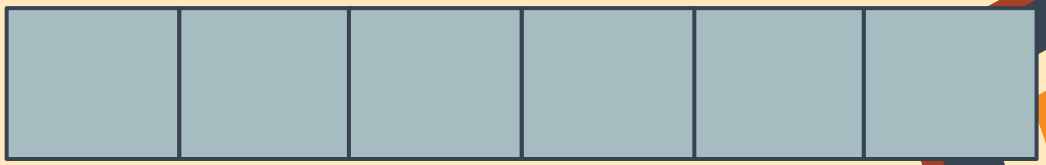
Object[] array={obj1,obj2,obj3};

Let’s create an array

Let’s create an array (of integers)!

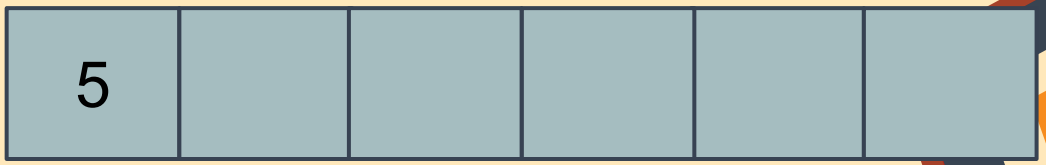
Let’s create an array (of integers)!

```
int[] arr = new int[6];
```



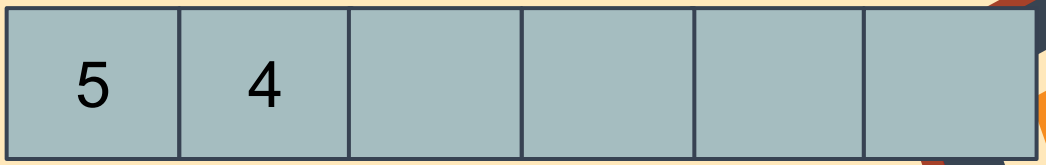
Let’s create an array (of integers)!

int[0] = 5;



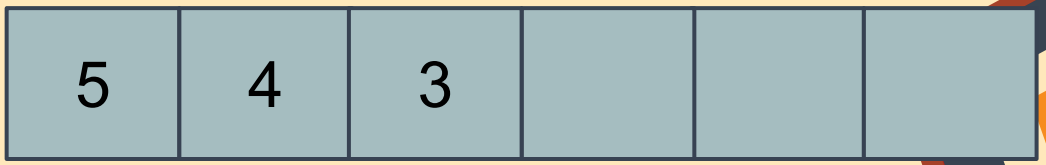
Let’s create an array (of integers)!

int[1] = 4;



Let’s create an array (of integers)!

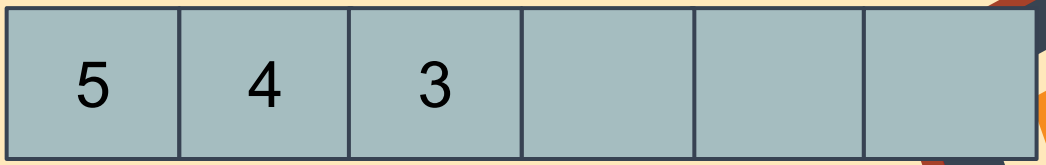
int[2] = 3;



Let’s create an array (of integers)!

System.out.println(arr[0]);

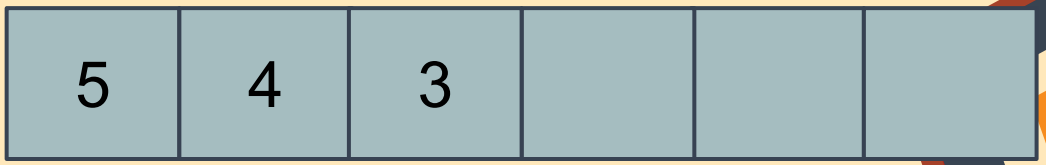
→ 5



Let’s create an array (of integers)!

System.out.println(arr[3]);

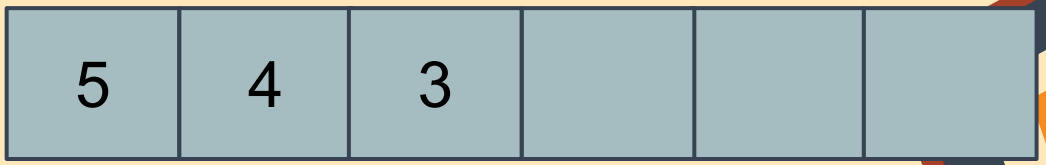
→ ERROR??? Maybe nullpointerexception?



Let’s create an array (of integers)!

System.out.println(arr[6]);

→ ERROR!!! OutOfBoundsException



Some cool array functions

- *Length of array*

  array.length; WARNING: Length Does Not Indicate The Number of Entries

- *Clone the array, returns a copy of the array* with the same type

  array.clone();

- *Get i-th index, i<array.length*

  array[i];

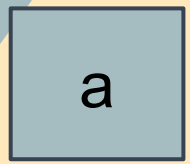
Let’s create an ArrayList (of strings)!

ArrayList<String> al = new ArrayList<String>();

→ QuickNote: If the data types in the lists are primitive (ie. int, boolean), then <e> part is not needed.

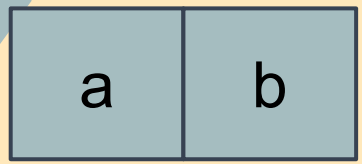
Let’s create an ArrayList (of strings)!

al.add(“a”);



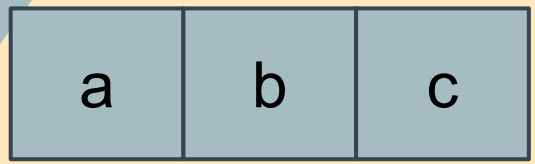
Let’s create an ArrayList (of strings)!

al.add(“b”);



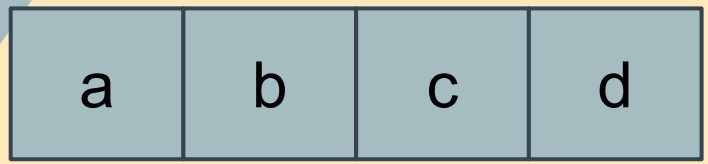
Let’s create an ArrayList (of strings)!

al.add(“c”);



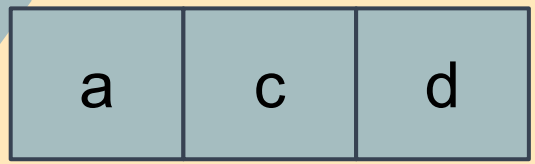
Let’s create an ArrayList (of strings)!

al.add(“d”);



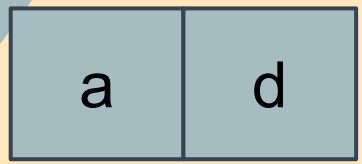
Let’s create an ArrayList (of strings)!

al.remove(“b”);



Let’s create an ArrayList (of strings)!

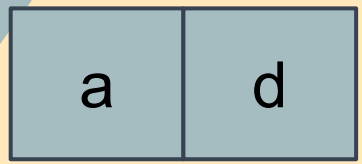
al.remove(1);



Let’s create an ArrayList (of strings)!

System.out.println(al.get(1));

→ d



Some cool arraylist functions

- *Length of arraylist*

  arraylist.size();
- *Set element at position index, return original value for element*

  arraylist.set(int idx, E element);
- *Get i-th index, i<arraylist.length*

  arraylist.get(i);

The 2 Dimensional Array

Did you know?

Each element of the array can also be another array!

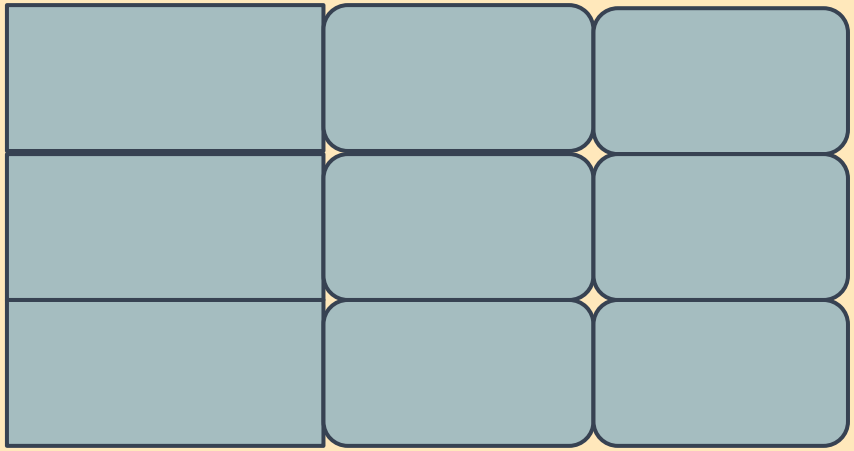
Declaration:

```
int[][] array=new int[size][size];
```

What is this Sorcery?

int[][] array=new int[3][3];

-Think of it as a 3x3 matrix



But that does not tell me anything...

int[][] array=new int[3][3];

It’s a 2-in one deal.

Think of it as a coordinate grid

The Best Approach: The Arrayception

int[][] array=new int[3][3];

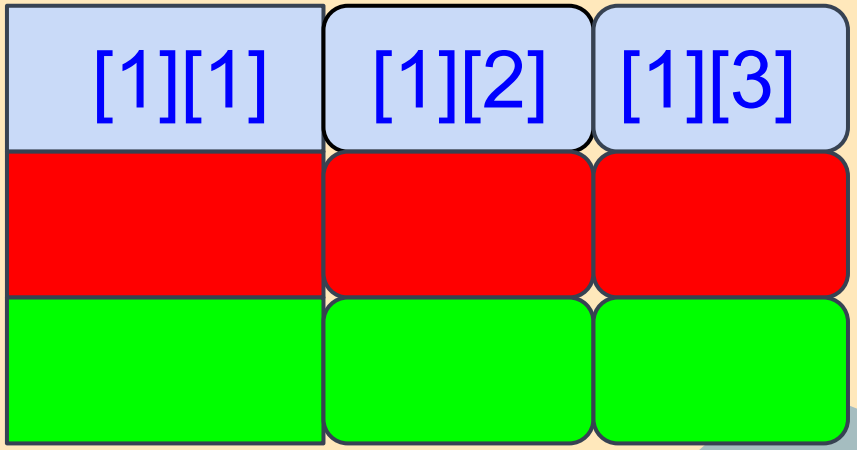
-It is an array that has int[] as its type: hence int[][]

-array[1-3][], the first indexes determines the index of array that contains all the other arrays.

A Means to an End

int[][] array=new int[3][3];

array[1][1-3], this dictates the index inside the array[1] in the array of int[]

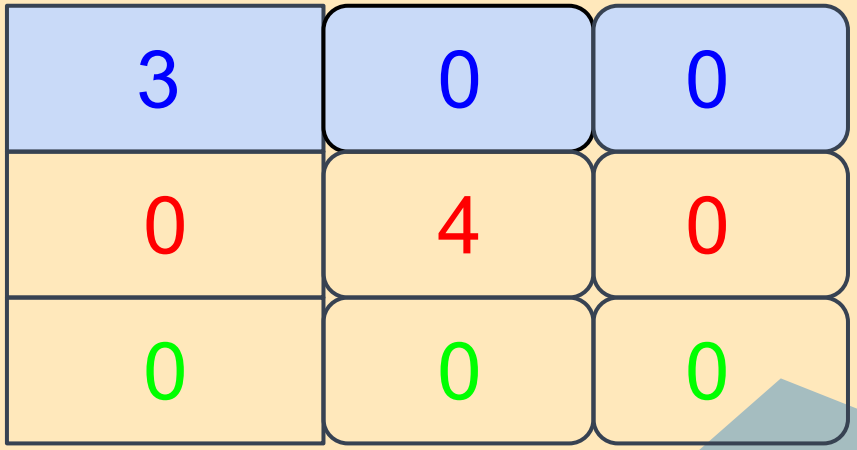


Seeing is Believing

int[][] array=new int[3][3];

-array[0][0]=3

-array[1][1]=4



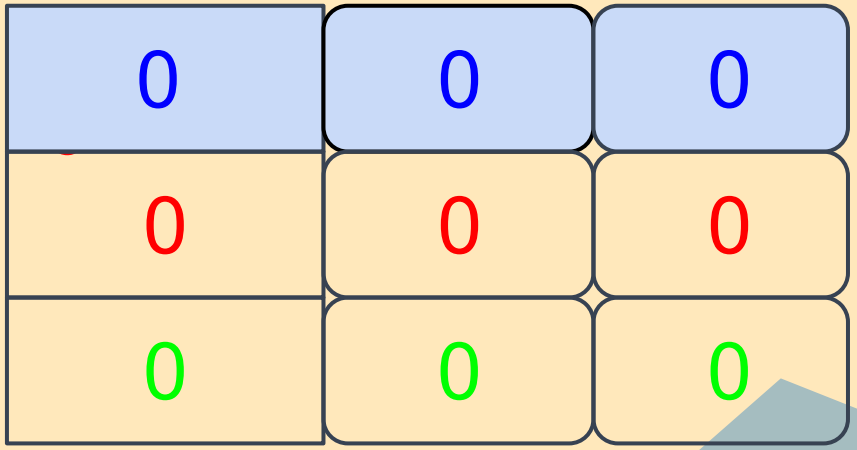
I need your help!!

int[][] array=new int[3][3];

-set array[1][2] to 6

-Set array[0][2] to 8

-set array[2][2] to 1092



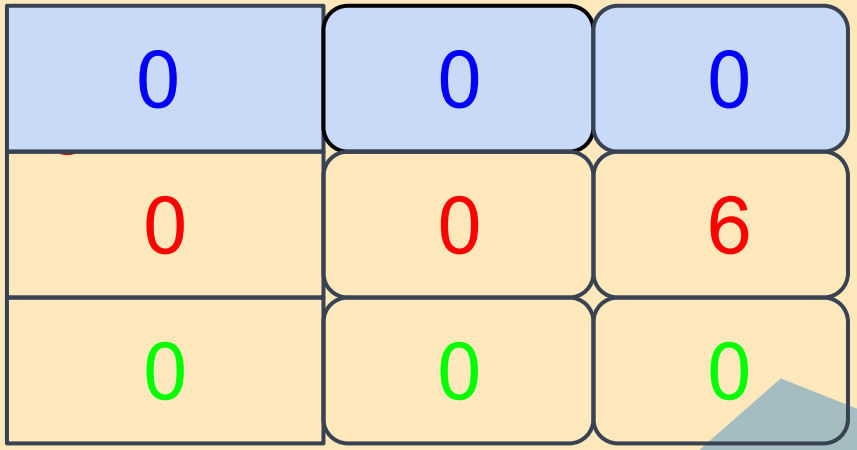
I need your help!!

int[][] array=new int[3][3];

-set array[1][2] to 6

-Set array[0][2] to 8

-set array[2][2] to 1092



I need your help!!

int[][] array=new int[3][3];

-set array[1][2] to 6

-Set array[0][2] to 8

-set array[2][2] to 1092

0	0	0
0	0	6
0	0	1092

I need your help!!

int[][] array=new int[3][3];

-set array[1][2] to 6

-Set array[0][2] to 8

-set array[2][2] to 1092

0	0	8
0	0	6
0	0	1092

2-D Arrays? How about 10-D?

Object[][][][][][][][][][] array=new Object[10][10][10][10][10][10][10][10][10][10];

Total Number of entries=10^10=10 billion entries

Take Away from This: You can make multidimensional array of any size.

Winners of the Century: Array vs ArrayLists

Arrays

→Doesn’t need to be imported

→Simpler coding

→Uses less memory

ArrayLists

→Dynamic Size, No limits! At least until your memory runs out

→Search Function!

→Easier to Manage

ArrayList<ArrayList<Integer>>

Losers of the Century: Array vs ArrayList

Arrays

→Fixed size

→No search feature so must iterate through entire array to find object

ArrayList

→No Primitive Type

int-Integer

boolean-Boolean

→Slow on occasions

→Imports