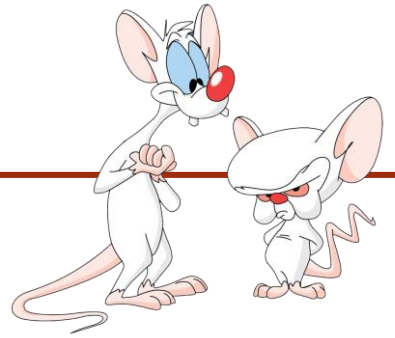
COMP 250
INTRODUCTION TO COMPUTER SCIENCE

Week 2: 2: Arrays

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Arrays
- Multidimensional arrays

ARRAYS

ARRAYS

- An array is like a container that holds a ***fixed number*** of values of the ***same type***. This values can be access using the same variable name.
  - The length of the array is established when the array is created and it cannot be changed.
  - All the elements of an array must be of the same type.
- The values in the array are called *elements.*
- The list of elements is indexed, the order matters.

ARRAYS – DECLARATION

To create an array, we first have to declare a variable with an array type:


```
type[] variable_name;
```

ARRAYS – DECLARATION

To create an array, we first have to declare a variable with an array type:


```
type[] variable_name;
```


The square brackets indicate that this is an array variable

ARRAYS – DECLARATION

To create an array, we first have to declare a variable with an array type:


```
type[] variable_name;
```


All the elements of the array must have this type.

The square brackets indicate that this is an array variable

ARRAYS – DECLARATION

- To create an array, we first have to declare a variable with an array type:


```
type[] variable_name;
```


- Examples:

```
String[] days;

int[] grades;

double[] heights;

String[] args;
```

ARRAYS – DECLARATION

- To create an array, we first have to declare a variable with an array type:


```
type[] variable_name;
```


- Examples:

```
String[] days;

int[] grades;

double[] heights;

String[] args;
```

NOTE: The declaration creates a variable, but **we still need to create the actual array object itself!**

ARRAYS – EXAMPLE

Here is a list of the days of the week:

0. Monday
1. Tuesday
2. Wednesday
3. Thursday
4. Friday
5. Saturday
6. Sunday

Note that we start counting at 0!!

We would like to use an array to store them all.

ARRAYS – INITIALIZATION METHOD 1

- **Using curly brackets**, we can assign values to all elements in the array at the same time as we declare the array:


```
String[] days = {"Monday", "Tuesday", "Wednesday",
                 "Thursday", "Friday", "Saturday", "Sunday"};
```


- `days` **is an array of** `String` **values of length 7.**
- Order matters!! What is the index of `"Tuesday"`?
- **This method does not work if you don't know the length or the values of the array before the program runs.**

ARRAYS – INITIALIZATION METHOD 2

- We can use the `new` operator to create an array of a certain size. We can then populate the entries one at a time, later in the program.

ARRAYS – INITIALIZATION METHOD 2

- We can use the **new** operator to create an array of a certain size. We can then populate the entries one at a time, later in the program.

- The following examples all create an array of `String` values with length 7.


```
String[] days = new String[7];
```

```
String[] days;
days = new String[7];
```

```
int numberOfDays = 7;
String[] days;
days = new String[numberOfDays];
```

ARRAYS – INITIALIZATION METHOD 2

- We can use the **new** operator to create an array of a certain size. We can then populate the entries one at a time, later in the program.

- After the creation, we can populate the entries


```
String[] days = new String[7];
days[1] = "Tuesday";
days[0] = "Monday";
days[6] = "Sunday";
days[5] = "Saturday";
days[2] = "Wednesday";
days[3] = "Thursday";
days[4] = "Friday";
```

DEFAULT VALUES

- As soon as we create an array object and initialize the variable, java assigns default values to each position in the array. Thus, if we don't assign any values to the array, it will have the values assigned by default.

- The default values are:
  - For numerical arrays: `0`
  - For String/reference type arrays: `null`
  - For char arrays: the character with ASCII value `0`
  - For boolean arrays: `false`.

SIZE OF AN ARRAY

- When we create an array of size `n`, `n` contiguous places in memory are reserved in order to store `n` values of the same type.

- The size of an array cannot be changed after it has been assigned.
- Any integer expression can be used for the size of an array as long as the value is nonnegative. If negative: **`NegativeArraySizeException`**.

ACCESSING ELEMENTS

- Elements in an array can be access using the name of the array variable and the index of the element inside square brackets.
  - `days[0] = "Sunday";`

    The value `"Sunday"` is assigned to `days` at index `0`
  - `String lastDay = days[6];`

    The value in `days` at index `6` is assigned to the variable `lastDay`
  - `days[0] = days[6];`

    The value in `days` at index `6` is assigned to `days` at index `0`.

- If we use an index that is negative or grater than the size of the array minus 1, the result is an **`ArrayIndexOutOfBoundsException`**.

ARRAYS AND LOOPS

- When we wanted to traverse a `String s`, we would loop through all the indices up to (and not including) `s.length()`
- Similarly, we can access the length of an array using `grades.length`


```
double sum = 0;
for (int i = 0;  i < grades.length; i++) {
   sum += grades[i];
}
double avg = sum/grades.length;
```

ARRAYS AND LOOPS

Note the difference!

- When we wanted to traverse a `String s`, we would loop through all the indices up to (and not including) `s.length()`
- Similarly, we can access the length of an array using `grades.length`


```
double sum = 0;
for (int i = 0;  i < grades.length; i++) {
   sum += grades[i];
}
double avg = sum/grades.length;
```

COMMON MISTAKES

- **Forgetting to initialize the entries of your array.**


```
int[] a = new int[5];
System.out.println(a[0]);
```

- **Output:** 0

COMMON MISTAKES

- Referring to an index that doesn't exists.


```
String[] petNames = {"Rex", "Sparky", "Bilbo", "Small cat"};
System.out.println(petNames[5]);
System.out.println(petNames[-2]);
```

- **`ArrayIndexOutOfBounds`** exception

COMMON MISTAKES

- Trying to create an array with negative size.


```
int size = -3 % 5;
String[] petNames = new String[size];
```

- **`NegativeArraySizeException`** exception

PRINTING DIFFERENCES

```
public static void myMethod(int[] a) {
   for(int i=0; i<a.length; i++) {
      System.out.println(a[i+1] - a[i]);
   }
}
```


The above code is supposed to loop through an array and print the difference between adjacent elements.

It contains an error. What is it?

ARRAYS

HOW TO CHECK EQUALITY?

```
int[] x = {1,2,3};
int[] y = {1,2,3};
System.out.println(x==y);
```


- What do you think the program will print?

```
false
```

HOW TO CHECK EQUALITY?

```
int[] x = {1,2,3};
int[] y = {1,2,3};
System.out.println(x.equals(y));
```


- **What do you think the program will print?**

```
false
```

HOW TO CHECK EQUALITY?

```
int[] x = {1,2,3};
int[] y = {1,2,3};
System.out.println(Arrays.equals(x,y));
```


- This compares the content of the two arrays.
- To use it, you need to import `java.util.Arrays;`

```
true
```

THE ARRAYS CLASS

- `Arrays` contains methods for manipulating arrays.
- `Arrays` is part of the `java.util` package.
- Before using it is useful to import it.
- https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

THE IMPORT STATEMENT

- The import statement for the `Arrays` class:


```
import java.util.Arrays;
```


- This statement allows us to refer to `Arrays` in the program without having to refer to the full package name (`java.util.Arrays`). It makes sure that the compiler knows which class we are referring to.

- All import statements are at the beginning of the file, before the class definition.

java.lang

- Exceptions to the import rule: all classes in the `java.lang` are imported by default.

- Remember the `Math` class? It is part of the `java.lang` package, therefore we did not need to import it in order to use it.

OTHER USEFUL METHODS?

In the Arrays class you can find:

- `Arrays.equals(x,y)`

  It returns a Boolean value indicating whether the content of the two arrays is the same.

- `Arrays.toString(x)`

  It returns the content of an array as a String.

- `Arrays.sort(x)`

  It sorts the input array in increasing order.

MULTIDIMENSIONAL ARRAYS

ARRAYS

- We have seen arrays as object with a fixed length containing elements of the same type.
- What if the elements where of array-type?
- How do we create arrays of arrays?

2D ARRAYS – HOW TO CREATE THEM

There are 3 ways to create arrays of arrays:


```
1.  int[][] numbers = {{1}, {1, 2}, {1, 2, 3}};

2.  int[][] numbers = new int[3][2];

3.  int[][] numbers = new int[3][];
```

2D ARRAYS – HOW TO CREATE THEM

```
int[][] numbers = {{1}, {1, 2, 3}, {1, 2}};
```


This creates an array of length 3, containing integer arrays. The first element is an array of length 1 (with value 1), the second an array of length 3 (with values 1, 2, 3), and the third an array of length 2 (with values 1, 2).

2D ARRAYS – HOW TO CREATE THEM

```
int[][] numbers = new int[3][2];
```


This creates an array of length 3. Each element is an integer array of length 2. You can think of it as a 3 by 2 matrix.

All the elements of the integer arrays are initialized by default to 0 (since they are int).

2D ARRAYS – HOW TO CREATE THEM

```
int[][] numbers = new int[3][];
```


This creates an array of length 3. Each element ***will be*** an integer array. For the moment the elements are initialized by default to `null`.

To populate the element of `numbers` we need to create 3 integer arrays and assign them to the elements of `numbers` one at a time.

2D ARRAYS – TWO EQUIVALENT ARRAYS

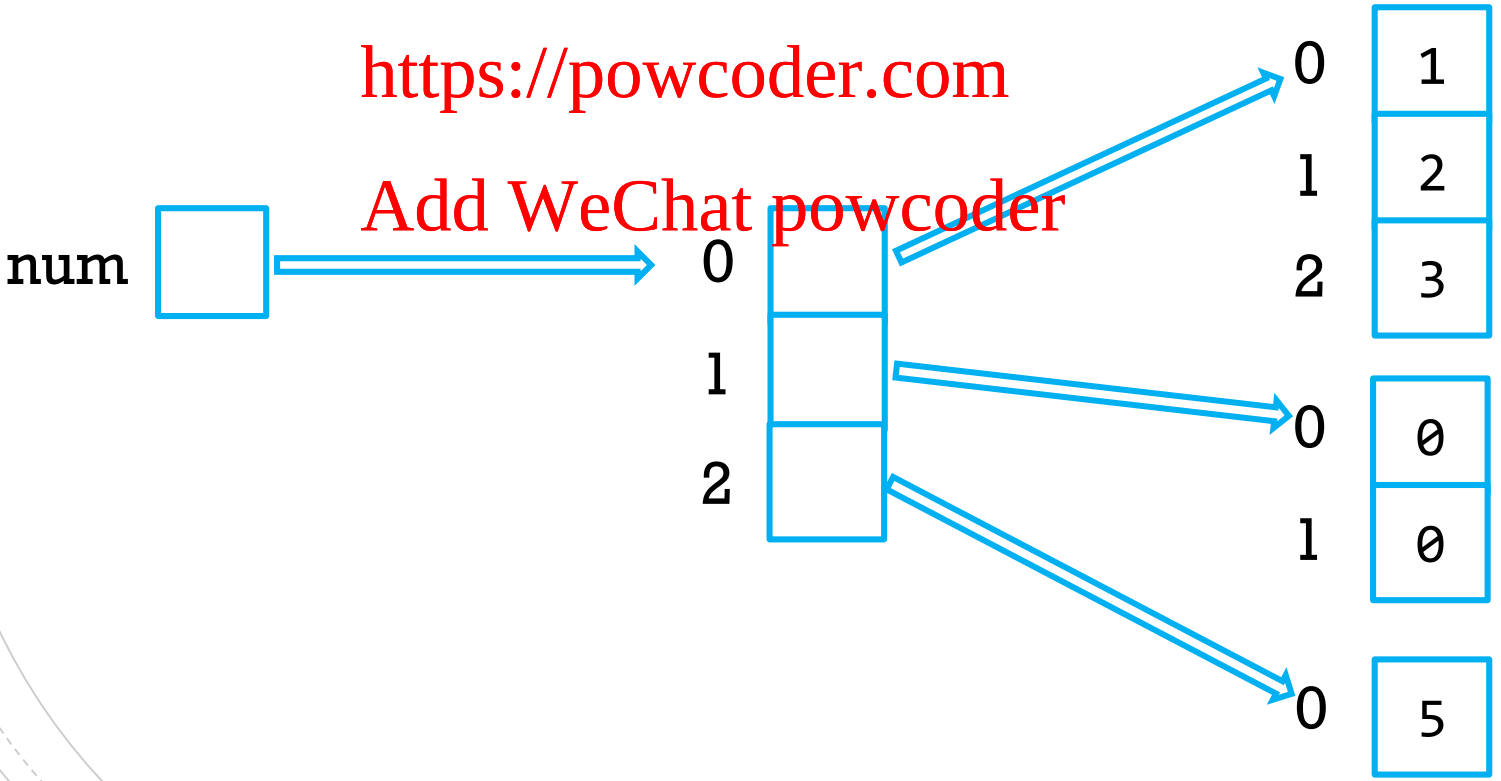
```
int[][] numbers = new int[3][];

int[] first = {1, 2, 3};

int[] second = new int[2];

numbers[0] = first;

numbers[1] = second;

numbers[2] = new int[1];

numbers[2][0] = 5;
```



```
int[][] num = {{1, 2, 3},{0, 0},{5}};
```

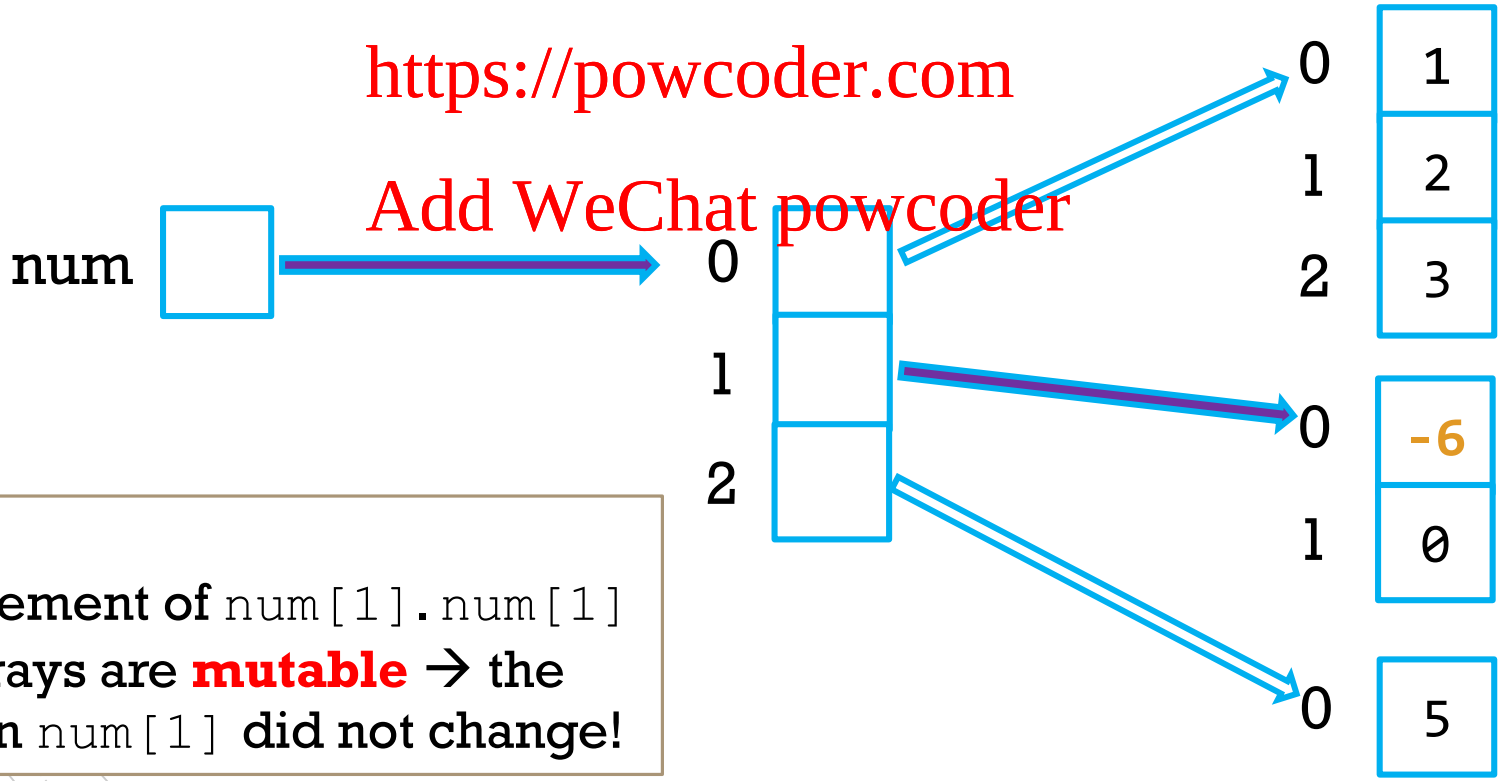
2D ARRAYS

```
int[][] num = {{1, 2, 3},{0, 0},{5}};
```

2D ARRAYS

```
int[][] num = {{1, 2, 3},{0, 0},{5}};
num[1][0] = -6
```

Note:

- We changed an element of `num[1].num[1]` is an array and arrays are **mutable** → the reference stored in `num[1]` did not change!

NOTE ON EQUALITY

- If you try to use `Arrays.equals(array1, array2)` on two arrays of arrays, it will not compare the elements of the inner arrays.

- If you want to compare the elements of the inner arrays you should use

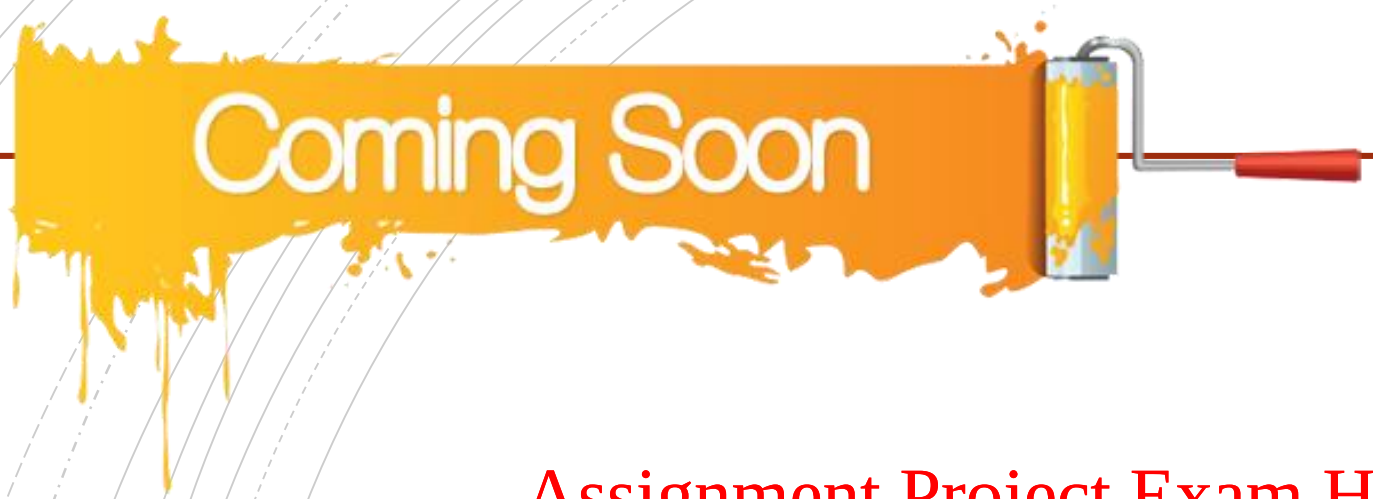
```
Arrays.deepEquals(array1, array2)
```

MULTIDIMENSIONAL ARRAYS

- You can create arrays with as many dimensions as you like.


```
int[][][] matrix3D = new int[3][3][3];

String[][][][] string4D = new String[2][][][];
```

- In practice, anything higher than 3D is rarely used.



In the next video we will be talking about reference types and the class `Random`.