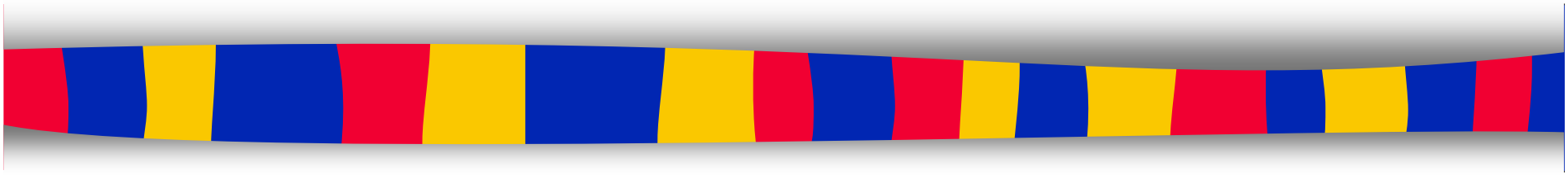
COMP-248
Object Oriented Programming I



Arrays of Primitive Types

By Emad Shihab, PhD, Fall 2015,
Parts of the slides are taken from Prof. L. Kosseim
Adapted for Section EE by S. Ghaderpanah, Fall 2015

Next:

1. **Arrays**
2. Sorting
3. "For each" loop
4. Multidimensional arrays

1- Arrays

Problem:

read 200 marks and compute how many are < the average…

```
Please enter mark nb 1:   80
Please enter mark nb 2:   65.5
…
Please enter mark nb 200:  68
The average is:  65.3
117 students have a mark higher than the average
```

Output

need to store 200 variables !!!

Solution:

use an array!

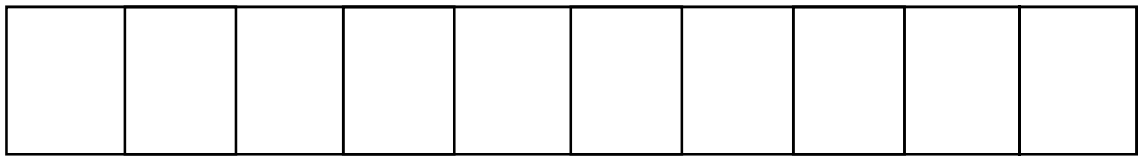
an array is an object that help us organize large amounts of information

Arrays

An *array* is an ordered list of elements of the same type

The entire array has a single name

Each value has a numeric *index*



This array holds 10 values that are indexed from 0 to 9

An array of size n is indexed from 0 to n-1

Arrays

The elements of an array can be:

- a primitive type or
- an object reference (we'll see this later)

Declaring and creating arrays

declaring the reference:

syntax: `type_of_elements[] name_of_array;`

```
int[] scores;        double[] marks;
char[] vowels;    String[] sentence;
```

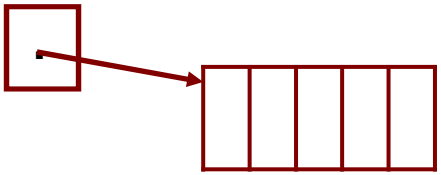
creating the elements:

syntax: `name_of_array = new type_of_elements[size];`

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

declaration + creation:

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



Declaring and creating arrays

size of the array:

must be an integer expression (constant or variable)

```
double[] price;
int nbItems = keyboard.readInt();
price = new double[nbItems];
```

array initialization:

every element is initialized to zero

int, double --> 0

boolean --> **false**

references --> **null**

Initializer lists

we can declare and initialize an array in one step

```
int[] units = {147, 323, 89, 933, 540, 269, 298, 476};
char[] letterGrades = {'A', 'B', 'C', 'D', 'F'};
```

Note:

- no size value is specified (size = nb of elements specified)
- the **new** operator is not used

Just checking …

Which of the following correctly initializes the indexed variables of an array named **myDoubles**?

A. `double myDoubles[double] = {0.0,1.0,1.5,2.0,2.5};`

B. `double myDoubles[5] = new double(0.0,1.0,1.5,2.0,2.5);`

C. `double[] myDoubles = {0.0,1.0,1.5,2.0,2.5};`

D. `array myDoubles[double] = {0.0,1.0,1.5,2.0,2.5};`

E. All of the above are valid

Just checking …

Given the declaration

```
int[] alpha = new int[75];
```

the valid range of index values for alpha is:

A. 0 through 75
B. 0 through 74
C. 1 through 75
D. 1 through 74
E. 1 through 76

Access to an element

syntax: **nameOfArray[indexOfElement]**

```
double[] scores = new double[10];
scores[2] = 55.5;
scores[0] = scores[2] + 2;
 double mean = (scores[0] + scores[2])/2;
System.out.print(mean);
```

56.5

Output

How can we assign each element one by one?

Automatic bounds checking

the index must be [0 … length-1]

The Java interpreter checks for you…

if an array index is out of bounds --> `ArrayIndexOutOfBoundsException`

```
double [] codes = new double[100];
int count = 100;
System.out.println(codes[count]);
```

```
for (int index=0; index <= 100; index++)
   codes[index] = index*50 + epsilon;
```

Just checking …

Consider the following array:

myArray[0]	7
myArray[1]	9
myArray[2]	-3
myArray[3]	6
myArray[4]	1
myArray[5]	-1

What is the value of

`myArray[myArray[1] – myArray[0]]`

A. 2
B. 9
C. -3
D. 6
E. 7

Just checking …

Given the declarations

```
int[] status = new int[10];
int i;
```

Which of the following loops correctly fills the array **`status`** with 1s?

A. `for (i=0; i<=10; i++)  status[i] = 1;`

B. `for (i=0; i<10; i++)  status[i] = 1;`

C. `for (i=1; i<=10; i++)  status[i] = 1;`

D. `for (i=1; i<10; i++)  status[i] = 1;`

E. `for (i=1; i<=11; i++)  status[i] = 1;`

The length instance variable (p. 344)

an array is an object

length is a public constant (an attribute) that gives the nb of elements in the array

```
int[] score = new int [10];
System.out.print(scores.length);
```

Example 1: nb of marks > average

```
How many students? 10
Please enter mark nb 1:   80
Please enter mark nb 2:   65.5
…
Please enter mark nb 10:  68
The average is:  65.3
4 students have a mark higher than the average
```

Output

Variables needed?

Algorithm?

Code

```
double sum = 0; double avg = 0.0; int students_abv_avg =0;
Scanner keyboard = new Scanner(System.in);
System.out.println("How many students?");
int students = keyboard.nextInt();

double [] scores = new double[students];

for (int i = 0; i < students; i++)
{
        System.out.println("Enter a student grade:");
        scores[i] = keyboard.nextDouble();
        sum += scores[i];
}

avg = sum/students;

for (int j = 0; j < students; j++)
{
        if (scores[j]> avg)
                students_abv_avg++;
}
System.out.println("The average is:" + avg);
System.out.println("The number of students above avg is:" + students_abv_avg);
```

Example 2: Difference from max

```
Enter 5 scores:
80 99.9 75 100 85.5
The highest score is 100
The scores are:
80.0 differs from max by 20
…
```

Variables needed?

Algorithm?

Code

```
Scanner keyboard = new Scanner(System.in);
double[] score = new double[5];
int index;
double max;

System.out.println("Enter " + score.length + " scores:");
score[0] = keyboard.nextDouble( );
max = score[0];
for (index = 1; index < score.length; index++)
{
    score[index] = keyboard.nextDouble( );
    if (score[index] > max)
        max = score[index];
//max is the largest of the values score[0],..., score[index].
}

System.out.println("The highest score is " + max);
System.out.println("The scores are:");
for (index = 0; index < score.length; index++)
    System.out.println(score[index] + " differs from max by "
                       + (max - score[index]));
```

Next:

1. Arrays
2. Sorting
3. For each loop
4. Multidimensional arrays