Handout 10

Arrays.

A data structure for storing homogenous data sets (i.e. sets of same type)
At first, we will consider arrays of primitive data values, then arrays of objects.

Problem: Find how many grades are above the average:
Ask the user to enter how many grades there are. Then ask them to enter the grades one by one. Compute the average grade, and how many grades are above the average.

=> need to store all grades => need an array!

Syntax for declaring an array:

```
<typename>[] Array_Name = new  <typename>[Length];
```

Examples:

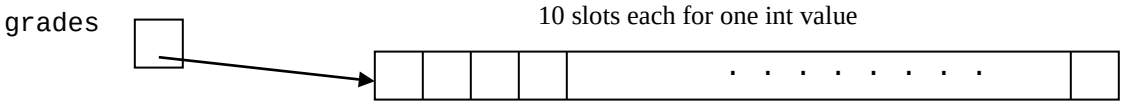
1.
```
int[] grades = new int [10];
```

declare variable of type int array

create the array of 10 int elements.

`array variable` it's a <u>reference</u> variable, i.e. stores a reference to the array:

Diagrammatically:



2.
```
// read the size of the grades array from user
Scanner kb = new Scanner (System.in);
int numGrades;
System.out.println("Please enter how many grades?");
numGrades = kb.nextInt();

// now create the array
int[] grades = new int [numGrades];
```

Array indexing and length:

1. Refer to a specific element in the array (also called *an indexed variable*) by using a subscript (a.k.a. index) in [ ].

Array subscripts use zero-based numbering.
Example: grade[0] –refers to the first element of the array, grade[1] – the second, etc…

***Index* - also called a *subscript* - must be an int, or an expression that evaluates to an int**

2. To get the length of array add `.length` to the name, e.g.

```
for (int i = 0; i < grades.length; i++){
     System.out.println (grades[i]);
}
```

(Notice the absence of parenthesis after the word length)

The length attribute is established in the declaration and **cannot be changed.**

Array Initialization:

1. Initialization (and other kinds of loop processing) can be done using a loop.
   Loop counter is often used as an array subscript(index)

```
int i;//loop counter/array index
int[] a = new int[10];

for(i = 0; i < a.length; i++)
   a[i] = 0;
```

2. Array elements can also be initialized in the declaration statement by putting a comma-separated list in braces as shown on the example below.
The length of an array is automatically determined when the values are explicitly initialized in the declaration
Example:

```
double[] reading = {5.1, 3.02, 9.65};
System.out.println(readings.length);
```

Q: what will be displayed?

Practice problems:

1. Write a program that
   a. reads in an average monthly rainfall for the year in Boston for each month (12 values),
   b. reads in the actual monthly rainfall for the past year (12 values)
   c. outputs, for each month, how much difference was there between the actual rainfall and the average, for each month (12 values).

   Example:

Input average rainfall :	2.3	14.5	23.6	17.5 …
Input actual rainfall :	1.1	12.3	25.0	17.0 …
Difference:	-1.2	-2.2	2	-0.5 …

2. Write a program that will let the user enter a list of names and will print out
   a. The number of different names in the list
   b. A set of all distinct names from the list.

3. Design and implement a program that creates a histogram that depicts the frequency distribution of a set of values. The program should read an arbitrary number of integers in the range 1 to 100 inclusive (terminated by –1) then produce a chart similar to one below that indicates how many input values fell in the range 1-10, 11-20, and so on. Print one asterisk for each value entered.

```
 1  – 10     | ****
11 –  20     | **
21 -  30     | ***************
31 -  40     | *********
41 -  50     | *****
51 –  60     | *******
61 –  70     | ***
71 -  80     | ***************
81 -  90     | *********
91 -  100    | *****
```

SUBTLE POINT:

Array variables are storing references to the actual arrays, so comparison (==) and assignment (=) operators compare the references and not the content of the arrays:

Example

```
int[] a = new int[3];
int[] b = new int[3];

for(int i; i < a.length; i++)
   a[i] = i;

b = a;

System.out.println(a[2] + " " + b[2]);
a[2] = 10;
System.out.println(a[2] + " " + b[2]);
```

The output for this code will be:

```
2 2
10 10
```

Same subtlety in comparisons (==):

```
int[] a = new int[3];
int[] b = new int[3];

for(int i=0; i < a.length; i++)
   a[i] = i;

for(int i=0; i < b.length; i++)
   b[i] = i;

if(b == a)
   System.out.println("a equals b");
else
 System.out.println("a does not equal b");
```

The output for this code will be: `a does not equal b`