Tables and Arrays

- An array is a collection of data items of the same type

  For example
  - A collection of integers
  - A collection of characters
  - A collection of strings
  - A collection of arrays
  - A collection of Objects

Creating an array in Java - Syntax

<data type>[] <array name> = new <data type>[<array size>];

An example:

int[] Mark = new int[5];

The preceding declaration will create an array (named Mark) that holds 5 integers!

Index	0	1	2	3	4
Mark	<int>	<int>	<int>	<int>	<int>

Entering values into an array

```
Mark[0]=85;
Mark[1]=65;
Mark[2]=73;
Mark[3]=78;
Mark[4]=80;
```

Our array now looks like this!!!

Index	0	1	2	3	4
Mark	85	65	73	78	80

Can we enter values another way?

- Yes, we can declare an array and enter the values at the declaration

```
int[] Mark = {85, 65, 73, 78, 80};
```

This creates an array of 5 integers (there are, after all only 5 integers in the {}) with values already entered

Index	0	1	2	3	4
Mark	85	65	73	78	80

The size of an array

- To find out how many values your array will hold we use the command <array name>.length which returns an integer indicating the number of values in your array

- e.g.
  System.out.println(**Mark.length**);

OUTPUT
5

Loops and Arrays

- Rather than accessing each element individually

e.g.
```
System.out.println(Mark[0]);
System.out.println(Mark[1]);
System.out.println(Mark[2]);
System.out.println(Mark[3]);
System.out.println(Mark[4]);
```

we can use a repetition construct to access each element of the array quickly.

For Example:

```
int[] Mark = {85, 65, 73, 78, 80};

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

OUTPUT	MEMORY