How do we create a one dimensional array?

One dimensional arrays were introduced during CS141 lectures number 16 and 17. Here we will focus on what you need to do in order to create an array of integers in your program (creating a chain of memory boxes).
We will then go on to show how an array is passed in to a method and also how an array is created in a method and how the array reference is returned by the method.

In general an array declaration looks like this:

```
type[] arrayName = new type[ length ];
```

This determines both the type of information in each memory box and the number of memory boxes. Once an array has been constructed, the number of boxes it has will **not** change. You can also separate the **array reference declaration** from the memory block assignment

```
      type[] arrayName;   // declaring an array reference

      arrayName  =  new  type[  length  ];  //create  boxes
linked to the array reference variable
```

Array creation continued

You can declare, construct, and initialise the array all in one single statement:

```
int[] data = {23,38,14,-3,0,14,9,103,0,-56};
```

This line of code declares an **array of type int reference variable** which is named data. **(step 1, Left hand side)**

Then it constructs an int array of 10 memory boxes (indexed 0…9). **(step 2, Right hand side)**

Finally, it puts the designated values into the boxes. The first value in the **list** goes into slot indexed [0], the second value goes in to slot indexed [1], and so on. (So in this example, `data[0]` gets the 23.)

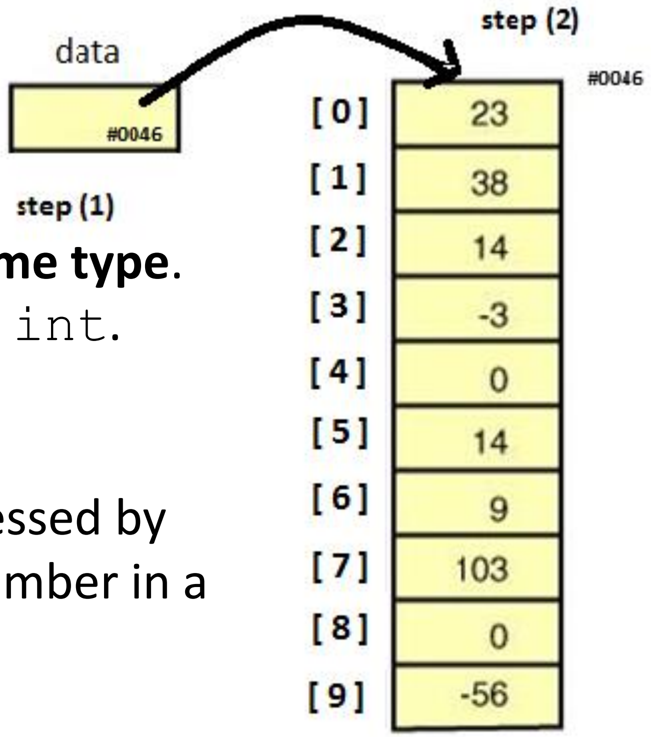
The compiler will count the values in the list and make that many boxes.

Read the Java declaration type `int[]`, as a reference variable type that will be linked to a chain of memory boxes eventually containing int values.

Recap on the structure of an array

An array is made out of a contiguous (individual boxes are in contact) block of memory which is divided into a number of boxes.

```
int[] data = {23,38,14,-3,0,14,9,103,0,-56};
```



Each box holds a value, and **all the values are of the same type**. In the example array here, each box holds value of type `int`. The name (identifier) of this array is `data`.

The slots are indexed 0 through 9. Each box can be accessed by using its **index** . (the index is like an individual house number in a housing estate)

For example, **data[0]** is the box which is indexed by zero (which contains the value **23**). **data[5]** is the box which is indexed by the number 5 (containing the value **14**).

The individual memory **boxes** that are part of an array, are commonly referred to by computer programmers as:

array element(s)

or as **element(s) of an array**.

You will hear me use these terms.

What is the array ??????

```
int[] data = {23,38,14,-3,0,14,9,103,0,-56};
```

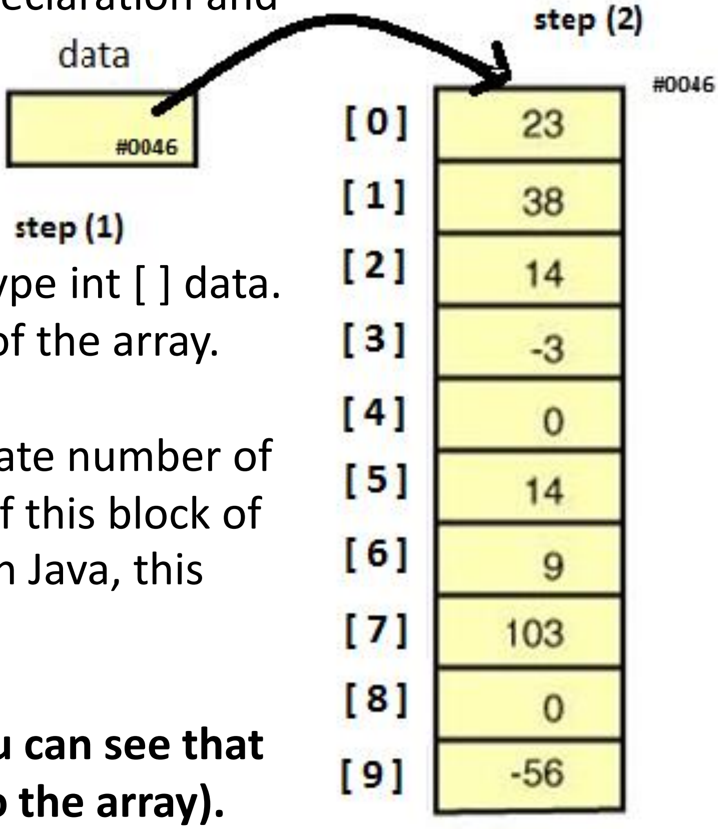
You can only create an array in this way when the array declaration and the list appear on the same line.

There are two steps to the setting up of an array.

Step (1) Declares an array reference of the appropriate type int [ ] data. This will **eventually** point to the block of memory boxes of the array.

Step (2) The list is read by the compiler and the appropriate number of memory boxes is created. The starting point or address of this block of memory is stored in the array reference variable `data`. In Java, this address information is hidden from the programmer

NB: We usually say that this array is named data but you can see that data actually holds the address of the array (or points to the array).



```
int data[] = {23,38,14,-3,0,14,9,103,0,-56}; // this is also correct
```

Important facts about one dimensional arrays?

The **length** of an array is how many boxes it has. An array of length N has boxes indexed 0…(N-1)

E.g. In an array made up of 5 boxes, the indexes of these boxes are 0, 1, 2, 3, 4

Indexes must be an integer type. It is OK to have spaces around the index of an array.

It is ***not*** *legal* to refer to a box in an array that does not exist:
Say that an array was declared as:

```
      int [ ] data = new int [10];
```

Here are some elements of this array, are they valid?

```
data[ -1 ] always illegal
data[ 10 ] illegal (given the above declaration)
data[ 1.5 ] always illegal
data[ 0 ] always OK
data[ 9 ] OK (given the above declaration)
```

Important facts about one dimensional arrays?

Now, when you get

```
Error line 17: ArrayIndexOutOfBoundsExceptionError
```

It means that you have overstepped the boundaries of the array, either with an index less than 0, or greater than N-1, where N is the length of the array.

When N is **10** as in the example below:

```
int [ ] data = new int [10];
```

Then the following attempts to access array boxes are **wrong and illegal**

```
data[ -1 ] always illegal

data[ 10 ] illegal (given the above declaration)
```

`Run time error: Array Index Out Of Bounds Exception Error`