Partially filled arrays

So far….

- We would declare an array that was exactly the right size to accommodate the data set that was being dealt with.

- What if there was a growing collection of data that we wanted to store in an array?
  - What if we needed our array to grow?

- If an array becomes full, the ideal solution would be to increase the size of the array.

- But!!! The size of an array is fixed at declaration!
  - E.g. int [] temp = new int [3];
    - Creates an array called temp of type array of int of **SIZE 3**

The solution

1) Create a new array that is larger than the original.
2) Copy all the elements of the array to the temporary one.
3) Set the original array reference to the temporary array.

For example:

```
int [] marks = {56, 76, 81}

//step 1
int [] temp = new int(marks.length+1)

// step 2
for (int c = 0; c<marks.length; c++){
        temp[c] = marks[c];
}

//step 3
marks = temp;
```

However…

- In the previous segment of code both ‘marks’ and ‘temp’ are referencing to the same array.
  - That is, if we modify anything in ‘temp’, the changes will also affect ‘marks’

Solution:

- Put the previous segment of code in a method.
  - Now ‘temp’ will be created in a method. When the method has completed compiling, ‘temp’ will be discarded.

For example:

```
public static int[] increase (int[] x){
        int [] temp = new int[x.length+1];
        for (int c = 0; c< x.length; c++){
                temp[c]= x[c];
        }
        return temp;
}

public static void main(String[] args) {
   int [] marks= {56,81,76};
   marks = increase (marks);
   marks = increase (marks);
}
```