Arrays

BACKGROUND

In some code you want to have several variables that represent similar values at the same time. Examine this piece of code:

```
int num1 = 4;
int num2 = 3;
int num3 = 9;
int num4 = 10;
num1++;
num2++;
num3++;
num4++;
```

You create four different variables that are all of the same type and then do the same manipulation of each variable. This works, but can be long, especially if you need to have a lot more than four variables or do a lot of the same operations to each variable. It'd be nice if you could group them all together, do all of these in a group. That's what arrays are for.

CREATING AN ARRAY

An array is a list of values of the same type. They use the same name and are usually treated as the same variable, but can still be individually accessed. There are two ways to create an array: one that doesn't give it starting values, and one that does. To create and array without values looks like this:

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

This should look very similar to all other variable creation, but there are some key differences. The first part (`int[]`) is the variable type, in this case an `int` array. You can create an array of any variable type, so this first part could be `double[]` or `String[]`. Square brackets are always used to indicate that it is not just an individual variable of that type, but an array of them. Then comes the variable name, which follows the same rules as any other variable name. Although it follows the same rules, it is common to put something in the name that helps to indicate that is is an array (like arr, array or list). After the equals sign you have the word `new`. That will just always be there when creating more complicated variables in Java. Then we have `int` (or `double` or `String`) with square brackets again. The last part that is important here is the number. This number is the length of the array. This will not change over the life of your program. So your list will always be the same length. In this example, it is 10 items long.

To create an array with values looks like this:

```
int[] numList = {1, 4, 6, 9};
```

The first part of this looks the same as before, but after the equals you have the starting values inside curly braces (`{}`). So this ends up being a 4 item list with the starting values of 1, 4, 6 and 9. The values can change over the life of the program, but the length of the list will not.

USING AN ARRAY

Using individual values

Any individual item in the list can be accessed using this syntax

```
int[] numList = {1, 4, 6, 9};

System.out.println(numList[0]);
```

This code will print 1. The first item in the list uses index 0, so if I printed `numList[2]`, it would print 6. Aside from the slightly different syntax, an item in an array can be accessed just like any other variable. You can do math with them; you can print them out; anything you can with another variable of that type, you can do with a member of an array.

Loops

Using arrays with loops is common enough to warrant some special attention. Let's take a look at the example from the start using array and a loop

```
int[] numList = {4, 3, 9, 10};

for (int i = 0; i < numList.length; i ++) {

  numList[i]++;

}
```

There are a few things to note here. First of all, you'll see that this is a lot shorter, but it will still do the same thing and we could make it do a lot longer of a list and only change the first line where it creates the list. Next thing to notice is the `numList.length` part. You can automatically get the length of the array using this for every array. The last thing to note here is that when you are indexing into an array, you can use a variable of type `int`.

TRY IT OUT

Starter problems

1. Swap the ends of an array. So if the array starts out as {1, 2, 3, 4, 5}. It should end up as {5, 2, 3, 4, 1}.
2. Count the number of even numbers in an array. So if your array is {4, 6, 10, 3, 5, 4, 0, -2}. Your answer will be 6.
3. Create a "centered" average of an array of numbers, average the numbers in the array but ignore the highest and lowest value. So if you had {100, 0, 2, 4, 3}, your answer would be 3.

More complicated stuff

If you really want a challenge, you can try these, but they are a lot more involved and will take a lot more code.

1. Find a way to sort your array. If you can do this, find a way that requires fewer operations in your code.
2. Find a way to shuffle your array. This will require a bit of investigation into random number generation.
3. How would you play tic-tac-toe in code? Start with just having two players and drawing a board and letting both players alternate turns. If you want you can get into a computer playing. It may start just by playing randomly, but if you really want to get fancy with it, try making it smart.