

Arrays Review and Exercises

- So far, you have been working with variables that hold only one value. The integer variables you have set up have held only one number, and the string variables just one long string of text. An array is a way to hold more than one value at a time. It's like a list of items. Think of an array as the columns in a spreadsheet. You can have a spreadsheet with only one column, or lots of columns. :

Array_Values	
0	10
1	14
2	36
3	27
4	43
5	18

- Like a spreadsheet, arrays have a position number for each row. The positions in an array start at 0 and go up sequentially. Each position in the array can then hold a value. In the image above array position 0 is holding a value of 10, array position 1 is holding a value of 14, position 2 has a value of 36, and so on.

- To set up an array of number like that in the image above, you have to tell Java what kind of data is going in to your array (integers, strings, boolean values, etc). You then need to say how many positions the array has. You set them up like this:

```
int[ ] aryNums;
```

- The only difference between setting up a normal integer variable and an array is a pair of square brackets after the data type. The square brackets are enough to tell Java that you want to set up an array. The name of the array above is aryNums. Just like normal variables, you can call them almost anything you like (with the same exceptions we mentioned earlier).
- But this just tells Java that you want to set up an integer array. It doesn't say how many positions the array should hold. To do that, you have to set up a new array object:

```
aryNums = new int[6];
```

- You start with your array name, followed by the equals sign. After the equals sign, you need the Java keyword new, and then your data type again. After the data type come a pair of square brackets. In between the square brackets you need the size of the array. The size is how many positions the array should hold.
- If you prefer, you can put all that on one line:

```
int[ ] aryNums = new int[6];
```

- So we are telling Java to set up an array with 6 positions in it. After this line is executed, Java will assign default values for the array. Because we've set up an integer array, the default values for all 6 positions will be zero (0).
- To assign values to the various positions in an array, you do it in the normal way:

```
aryNums[0] = 10;
```

- Here, a value of 10 is being assigned to position 0 in the array called aryNums. Again, the square brackets are used to refer to each position. If you want to assign a value of 14 to array position 1, the code would be this:

```
aryNums[1] = 14;
```

- And to assign a value of 36 to array position 2, it's this:

```
aryNums[2] = 36;
```

- Don't forget, because arrays start at 0, the third position in an array has the index number 2.

- If you know what values are going to be in the array, you can set them up like this instead:

```
int[ ] aryNums = { 1, 2, 3, 4 };
```

- This method of setting up an array uses curly brackets after the equals sign. In between the curly brackets, you type out the values that the array will hold. The first value will then be position 0, the second value position 1, and so on.

```
String[ ] aryStrings = {"Autumn", "Spring", "Summer", "Winter"};
```

- To set up a boolean array you still need the new keyword:

```
boolean[ ] aryBools = new boolean[ ] {false, true, false, true};
```

- To get at the values held in your array, you type the name of the array followed by an array position in square brackets. Like this:

```
System.out.println( aryNums[2] );
```

- If you know what values are going to be in the array, you can set them up like this instead:

```
int[ ] aryNums = { 1, 2, 3, 4 };
```

- This method of setting up an array uses curly brackets after the equals sign. In between the curly brackets, you type out the values that the array will hold. The first value will then be position 0, the second value position 1, and so on.

```
String[ ] aryStrings = {"Autumn", "Spring", "Summer", "Winter"};
```

- To set up a boolean array you still need the new keyword:

```
boolean[ ] aryBools = new boolean[ ] {false, true, false, true};
```

- To get at the values held in your array, you type the name of the array followed by an array position in square brackets. Like this:

```
System.out.println( aryNums[2] );
```

EXERCISE 1

- Start a new project and call it ArraysTest. Enter the following:

```
public class ArraysTest {

    public static void main(String[] args) {

        int[] aryNums;

        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

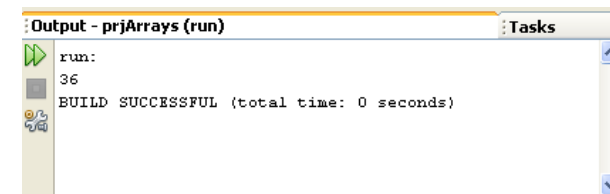
        System.out.println( aryNums[2] );

    }

}
```

EXERCISE 1 Cont

- When you run the programme you should see this output.



- Change the array position number in the print line from 2 to 5 and 18 should print out instead.

- Arrays come into their own with loops. As an example, imagine a lottery programme that has to assign the numbers 1 to 49 to positions in an array. Instead of typing a long list of array positions and values you can use a loop. Here's some code that does just that:

```
public class ArraysTest {

    public static void main(String[] args) {

        int[] lottery_numbers = new int[49];
        int i;

        for (i=0; i < lottery_numbers.length; i++) {
            lottery_numbers[i] = i + 1;
            System.out.println( lottery_numbers[i] );
        }

    }
}
```

- Instead of a hard-code value between the square brackets of the array name, we have the variable called `i`. This increases by 1 each time round the loop, remember. Each array position can then be accessed just by using the loop value. The value that is being assigned to each position is `i + 1`. So again, it's just the incremented loop value, this time with 1 added to it. Because the loop value is starting at 0, this will give you the numbers 1 to 49.
- The other line in the loop just prints out whatever value is in each array position.
- If you wanted, you could then write code to jumble up the numbers in the array. Once you have jumbled up the values, you could then take the first 6 and use them as the lottery numbers. Write another chunk of code that compares a user's 6 numbers with the winning numbers and you have a lottery programme!

- So we set up an array to hold 49 integer values. Then comes the loop code. Notice the end condition of the loop:

`i < lottery_numbers.length`

- Length** is a property of array objects that you can use to get the size of the array (how many positions it has). So this loop will keep going round and round while the value in the variable `i` is less than the size of the array.

- To assign values to each position in the array, we have this line:

`lottery_numbers[i] = i + 1;`

- Other inbuilt java methods allow you to sort your arrays. To use the sort method of arrays, you first need to reference a Java library called **Arrays**. You do this with the import statement.
- Try it with the array code from earlier, and add the following import statement:

```
import java.util.Arrays;

public class ArraysTest {

    public static void main(String[] args) {

        int[] aryNums;
        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

    }

}
```

- Now that you have imported the Arrays library, you can use the sort method.

```
import java.util.Arrays;

public class ArraysTest {

    Arrays.sort(aryNums);

    public static void main(String[] args) {

        int[] aryNums;
        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

        Arrays.sort(aryNums);

        int i;

        for (i=0; i < aryNums.length; i++) {
            System.out.println("num:" + aryNums[i]);
        }

    }

}
```

- You can place strings of text into arrays.

```
String[] aryString = new String[5];
aryString[0] = "This";
aryString[1] = "is";
aryString[2] = "a";
aryString[3] = "string";
aryString[4] = "array";
```

- Here's a loop that goes round all the positions in the array, printing out whatever is at each position:

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

- The loop goes round and round while the value in the variable called i is less than the length of the array called aryString.

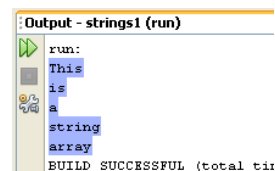
- You can place strings of text into arrays.

```
String[] aryString = new String[5];
aryString[0] = "This";
aryString[1] = "is";
aryString[2] = "a";
aryString[3] = "string";
aryString[4] = "array";
```

- Here's a loop that goes round all the positions in the array, printing out whatever is at each position:

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

- The loop goes round and round while the value in the variable called i is less than the length of the array called aryString.



```
Output - strings1 (run)
run:
This
is
a
string
array
BUILD SUCCESSFUL (total time: 0s)
```

- You can perform a sort on string arrays, just like you can with integers. But the sort is an alphabetical ascending one, meaning that "aa" will be before "ab".

```
import java.util.Arrays;

public class StringArrays {

    public static void main(String[] args) {

        String[] aryString = new String[5];
```

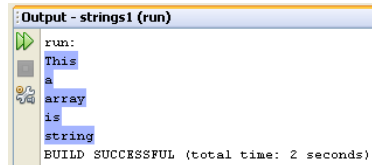
```
aryString[0] = "This";
aryString[1] = "is";
aryString[2] = "a";
aryString[3] = "string";
aryString[4] = "array";
```

```
Arrays.sort(aryString);
```

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

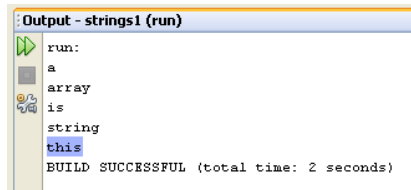
- However, Java uses Unicode characters to compare one letter in your string to another. This means that uppercase letter will come before lowercase ones.

- When the programme is run, the Output window will display the following:



```
Output - strings1 (run)
run:
This
a
array
is
string
BUILD SUCCESSFUL (total time: 2 seconds)
```

- Although we've sorted the array, the word "This" comes first. If this were an alphabetical sort, you'd expect the word "a" to come first." And it does if all the letters are lowercase. In your programming code, change the capital "T" of "This" to a lowercase "t". Now run your programme again. The Output window will now display the following:



```
Output - strings1 (run)
run:
a
array
is
string
this
BUILD SUCCESSFUL (total time: 2 seconds)
```

- Exercise 2**

Set up an array to hold the following values, and in this order: 23, 6, 47, 35, 2, 14. Write a programme to get the average of all 6 numbers. (You can use integers for this exercise, which will round down your answer.)

- Exercise 3**

Using the above values, have your programme print out the highest number in the array.

- Exercise 4**

Using the same array above, have your programme print out only the odd numbers.