Manipulating Arrays

by Sophia

WHAT'S COVERED

In this lesson, you will learn about various functions and methods that can be used to manipulate lists. You will also learn about the `Arrays.toString()` and `Arrays.sort()` methods. Specifically, this lesson covers:

1. Operators With Arrays

Values of single elements can be changed using the assignment operator (=).

1a. Assignment Operator (=)

Recall, the = operator (equal sign) is an **assignment operator** that is used to assign values to variables. Values can be assigned to an array as easily as it was to assign a value to a variable. Individual elements can be replaced in the array using the same operator as well. You must ensure to include the index number in square brackets. This will indicate which element is to be changed. Once an array has been declared, it is not possible to replace the whole array using only literal values in curly brackets.

TRY IT

Directions: Enter the following code and run it to see the element replacement.

```
class ArrayAssign {
  public static void main(String[] args) {
    String[] petList = {"dog", "cat", "fish"};
    System.out.print("Original array: ");
    System.out.println(petList[0] + ", " + petList[1] + ", " + petList[2]);
    petList[1] = "hamster";
    System.out.print("Modified array: ");
    System.out.println(petList[0] + ", " + petList[1] + ", " + petList[2]);
  }
}
```

The output should look like this:

```
Original array: dog, cat, fish
Modified array: dog, hamster, fish
```

REFLECT

As this small program shows, the assignment operator (=) works with individual elements in an array just like it does with any variable. Don't forget that the single equal sign is the assignment operator, while the double equal sign (==) is used to compare two values for equality.

In the example above, the element in index 1 was changed to hamster. This was the second element. Remember, the index starts at 0. To switch out dog for hamster, you would have needed to use petsList[0] with the index 0.

Next, consider changing the whole array using the assignment operator (=). As noted above, once an array has been declared, individual elements in the array can have their values changed using the single equal sign, but not the whole array. If you try to replace the whole array, it will generate an error.

TRY IT

Directions: Type the following code into the IDE in a file named ChangeWholeArray.java:

```
import java.util.Arrays;

class ChangeWholeArray {
  public static void main(String[] args) {
    // Original array contents
    int[] numbers = {1, 2, 3};
    System.out.println("Array contents: " + Arrays.toString(numbers));
    // Try to change whole array
    numbers = {4, 5, 6};
    System.out.println("New array contents: " + Arrays.toString(numbers));
  }
}
```

Trying to run this code will produce a number of problems, as seen below:

```
src/main/java/ChangeWholeArray.java:9: error: illegal start of expression
    numbers = {4, 5, 6};
              ^
src/main/java/ChangeWholeArray.java:9: error: not a statement
    numbers = {4, 5, 6};
               ^
src/main/java/ChangeWholeArray.java:9: error: ';' expected
    numbers = {4, 5, 6};
                ^
3 errors
error: compilation failed
```

REFLECT

The details of the error messages may seem confusing, but this small program makes clear that an entire array can't be modified just using an equal sign and appropriate values in curly brackets. Individual elements in an array can be changed, but not the whole array.

TRY IT

To change the contents of the array completely, use the `new` keyword to construct a whole new array object. Then, assign it using the single equal sign:

```java
import java.util.Arrays;

class ChangeWholeArray {
  public static void main(String[] args) {
    // Original array
    int[] numbers = {1, 2, 3};
    System.out.println("Array contents: " + Arrays.toString(numbers));
    // Assign whole new array using new and =
    numbers = new int[]{4, 5, 6};
    System.out.println("New array contents: " + Arrays.toString(numbers));
  }
}
```

Directions: Now use the following line:

EXAMPLE

```java
numbers = new int[]{4, 5, 6};
```

Doing so successfully changes the contents of the complete array to look like the output screen below:

```
Array contents: [1, 2, 3]
New array contents: [4, 5, 6]
```

When discussing variables earlier in the course, it was noted that the keyword final can be placed before the data type in a declaration. Doing this will result in a constant value. Unlike a variable, which can have its value changed, a **constant** declared with final cannot be changed from its initial value. The final keyword can be used when declaring a Java array, but it will not have the effect that one might expect. Since the individual elements in an array will always be mutable, using final will keep the whole array from being reassigned.

REFLECT

As we have seen, the assignment operator (=) by itself can't be used to change a whole array. Used in conjunction with the new keyword, though, the name of an existing array can be made to refer to a new array (unless the array has been declared final). We will learn more about the new keyword in the unit on Java classes.

Here is an example of how individual elements in an array declared as final can still be changed:

```
import java.util.Arrays;

class FinalArray {
  public static void main(String[] args) {
    // Original array - declared final (constant)
    final int[] numbers = {1, 2, 3};
    System.out.println("Array contents: " + Arrays.toString(numbers));
    // Assign new values to individual elements
    numbers[0] = 4;
    numbers[1] = 5;
    numbers[2] = 6;
    System.out.println("New array contents: " + Arrays.toString(numbers));
  }
}
```

The results will show that the individual array elements have been changed, as seen below (where they were changed separately):

```
Array contents: [1, 2, 3]
New array contents: [4, 5, 6]
```

TRY IT

Directions: Now, try to run the following code:

```
import java.util.Arrays;
```

```
class FinalArray {
  public static void main(String[] args) {
    // Original array - declared final (constant)
    final int[] numbers = {1, 2, 3};
    System.out.println("Array contents: " + Arrays.toString(numbers));
    // Assign new array
    numbers = new int[]{4, 5, 6};
    System.out.println("New array contents: " + Arrays.toString(numbers));
  }
}
```

Java attempts to assign a whole new array using = and new. However, this approach will not work with a final array. It will generate an error message, which indicates that the array is immutable because it was declared final, as seen below:

```
src/main/java/FinalArray.java:9: error: cannot assign a value to final variable numbers
    numbers = new int[]{4, 5, 6};
    ^
1 error
error: compilation failed
```

REFLECT

As the result produced by this code shows, it is important to keep in mind that elements in a Java array can always be modified (at least individually), even if the array has been declared `final`.

When an array is declared final, the array itself (the whole array as an object) is immutable. However, the individual elements in an array are always mutable.

TERMS TO KNOW

Assignment Operator
The assignment operator is a single equal sign (=) that is used to assign a value on the right side of the equal sign to an array element (or other variable) on the left of the equal sign.

Constant
A named value of a specific data type stored in memory that cannot be changed after it has been given an initial value.

2. Methods

Java includes a number of useful methods for working with arrays, in the Arrays utility class. In this section, we will discuss two of these methods. They include Arrays.toString() and Arrays.sort().

KEY CONCEPT

Keep in mind that when using these methods, that the Arrays class must be specified. Also consider that the name of the array to be converted to a String value or sorted is passed as an argument inside the parentheses.

2a. Arrays.toString() Method

The **Arrays.toString()** method is a useful utility for displaying the contents of an array.

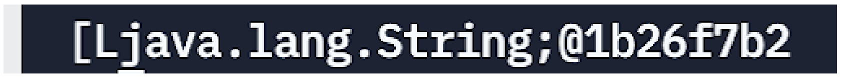
TRY IT

Directions: Try to print an array just using the name of the array (this leads to an odd result):

EXAMPLE

```
System.out.println(petList);
```

Doing so produces the unexpected result below:



```
[Ljava.lang.String;@1b26f7b2
```

This is internal code used by the JVM and not intended for human consumption. The `Array.toString()` method produces a representation of the array that is easier to read.

EXAMPLE A call to the code below produces a readable list of items in the array:

```
System.out.println(Arrays.toString(petList));
```

EXAMPLE Use of Arrays.toString() requires the following import statement near the top of the .java file:

```
import java.util.Arrays;
```

TRY IT

Directions: Practice using `Array.toString()` in the IDE, as follows:

```
import java.util.Arrays;

class PetsArray {
  public static void main(String[] args) {
    String[] petList = {"dog", "kitten", "fish"};
    petList[1] = "kitten";
    System.out.println(Arrays.toString(petList));
  }
}
```

Running this code appears to have worked much better:

```
[dog, kitten, fish]
```

REFLECT

Displaying the contents of an array to the user in square brackets might seem a bit unexpected in many cases, but Arrays.toString() is very valuable when developing and debugging code.

This method will be used often when discussing arrays in further tutorials.

2b. Arrays.sort() Method

The **Arrays.sort() method** arranges all of the elements in the array from the lowest to highest. If the values are strings, the result will be a list of strings in alphabetical order.

TRY IT

Directions: Try using the .sort() method on the petsList:

```
import java.util.Arrays;

class PetsArraySort {
  public static void main(String[] args) {
    String[] petsArray = {"dog", "cat", "fish", "rabbit", "hamster", "bird"};
    System.out.println("Original array: " + Arrays.toString(petsArray));
    Arrays.sort(petsArray);
    System.out.println("Sorted array: " + Arrays.toString(petsArray));
  }
}
```

The result should be as follows:

```
Original array: [dog, cat, fish, rabbit, hamster, bird]
Sorted array: [bird, cat, dog, fish, hamster, rabbit]
```

REFLECT

The output above shows that the order of the contents of the array has been changed, but the contents of the overall array haven't changed. The Arrays.sort() method will prove to be helpful in a number of contexts.

Bird to rabbit are all seen in alphabetical order. If the list contains numerical values, they are ordered from smallest to largest.

TRY IT

Directions: Try using the .sort() method again on a numerical list:

```
import java.util.Arrays;

class NumbersArraySort {
  public static void main(String[] args) {
    int[] numArray = {2, 45, 9, 17, 1, 2};
    System.out.println("Original array: " + Arrays.toString(numArray));
    Arrays.sort(numArray);
    System.out.println("Sorted array: " + Arrays.toString(numArray));
  }
}
```

The result should be as follows:

```
Original array: [2, 45, 9, 17, 1, 2]
Sorted array: [1, 2, 2, 9, 17, 45]
```

TERMS TO KNOW

Arrays.toString()
The Arrays.toString() method converts the contents of an array to String values that can be displayed on the screen.

Arrays.sort()
The Arrays.sort() method arranges all of the elements in an array from the lowest to highest. If the values are strings, the result will be a list of strings in alphabetical order.

SUMMARY

In this lesson, you learned about the **operators** and methods that can be used to manipulate arrays. Using these tools, we have learned various **methods and functions** that you can use to assign values to **arrays to string**, modify elements, print, and **sort arrays**.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from “Python for Everybody” By Dr. Charles R. Severance. Source py4e.com/html3/

TERMS TO KNOW

Arrays.sort()

The Arrays.sort() method arranges all of the elements in an array from the lowest to highest. If the values are strings, the result will be a list of strings in alphabetical order.

Arrays.toString()

The Arrays.toString() method converts the contents of an array to String values that can be displayed on the screen.

Assignment Operator

The assignment operator is a single equal sign (=) that is used to assign a value on the right side of the equal sign to an array element (or other variable) on the left of the equal sign.

Constant

A named value of a specific data type stored in memory that cannot be changed after it has been given an initial value.