# Object Oriented Programming – Java 1

Lecture 7

Arrays

Dr. Obuhuma James

**Description**

This is the last topic on the basic concepts of Java programming that can aid in the transition to object-oriented programming concepts. The topic covers arrays, which are a critical component of programming. Arrays aid in the storage of more than one item using one reference variable. The items are of the same data type and are stored the same distance apart in memory. The use of the for-each loop to access and display array elements is discussed as a way of concluding on the concept of looping introduced in the previous topic.

**Learning Outcomes**

By the end of this topic, you will be able to:

- Describe the concept of arrays as used in storage of items sets.
- Create arrays and assign data sets as array items.
- Access, use and display array items on the console.

**Overview of Arrays**

Arrays are named lists of data items of the same type [1]. An array is a set of data represented by a single variable name. In the case of tightly typed programming languages like Java, a given array must contain data of the same data type. This is because the definition of the array must specify the data type, just like for the case of variable declaration. On the other hand, loosely typed programming languages like PHP permit use of mixed data types within one array.

An array basically organizes data in such a manner that each piece of data is stored the same distance apart in memory, referenced using a single variable name, and has a fixed size in memory. Array indices are then used to distinguish the various data items stored in the array. In most programming languages, including Java, the default array numbering indices begin from 0, since the array index is an offset from the beginning of the location in memory where the array is stored, hence, the first element is located at an offset of 0 bytes.

**Declaring and Using Arrays**

In Java, arrays are declared the same way a simple variable is declared [1]. However, a pair of square brackets is inserted immediately after the data type.

Syntax

*datatype[] arrayname;*

Example

The following example statement creates an array called age of type integer for storing a set of student ages.

*int[] age;*

Part of array declaration requires specification of the array size in terms of the required memory size. Thus, the “new” keyword is then used together with the arrayname to set the array size.

Syntax

*arrayname = new datatype[sizeofarray];*

Example

To initialize the array age created in the previous example, the following statement should be used, where 10 specifies the size of the array.

*age = new int[10];*

The process above could be accomplished in one go by using the following approach

Syntax

*datatype[] arrayname = new datatype[sizeofarray];*

Example

*int[] age = new int[10];*

The above example creates an array called age of type integer then sets its size to 10. Because array elements are numbered from 0, you can comfortably use indices 0 through 9 when working with this array of 10 items as shown in Figure 1. The same concept applies to any size specified for an array.

| age[0] | age[1] | age[2] | age[3] | … | age[7] | age[8] | age[9] |
|---|---|---|---|---|---|---|---|

*Figure 1. Example Structure of an Array*

**Assigning Values to Array Items**

Working with any individual array element is hence treated like a single variable of the same type, only that the index of the item must be clearly known and specified. Thus, the following is the syntax for assigning values to array elements.

Syntax

*arrayname[index] = value;*

Example

The following examples shows how age value 32 could be assigned to position 3 of the array age.

*age[2] = 32;*

Notice the fact that the array index 2 is used to assign a value to position 3 since array numbering begins from 0. The same concept should be used to assign values to all other indices of the array.

Values can also be assigned to array elements at the point of creation. Such an approach hence does not require specification of the size of the array since the size is automatically determined based on the number of items assigned to the array.

Syntax

*datatype[] arrayname = {values1, value2, value3, … , valuen};*

Example

*int[] age = {23, 12, 32, 20, 13, 18, 16, 19, 14, 11};*

The above statement declares an array called age and assigns a set of 10 age values to the age. This means that the array will end up being of size 10 based on the number of items that have been initialized to it at the point of creation.

**Accessing Array Items**

To access array elements, the array name and index of the item being targeted must be known upfront.

Syntax

*arrayname[index];*

For instance, age[2], targets to access the item at position 3 in the array called age. Thus, to display the item in position 3 on the console, the System.out.print() statement has to be used together with the age[2] statement as follows.

Example

*System.out.print(age[2]);*

The same approach can be used to access and/or display any item at any position in the array. Similarly, items accessed from the array can equally be subjected to any operations, just like the case of data stored in variables.

**Accessing Array Items Using the for-each Loop**

The for-each is used to loop through elements in an array. For each iteration of a for-each loop, the loop moves to the next element in the given array. Unlike the previous looping statements, the for-each loop does not require a loop counter. However, a new variable for holding the items in the array has to be specified and linked to the array name using a full colon (:) within a set of parentheses following the for keyword.

Syntax

*for(datatype variablename: arrayname){*

*Statement(s) to be executed;*

*}*

There are two critical variables required in the for-each loop. The first variable should refer to the new variable to hold data items to be retrieved from the array while the second variable following the full colon (:) should be the variable representing the array name.

Example

Consider the example in Figure 2 that creates an array of continents. A demonstration of how the for-each loop can be used to access and display the three continents stored in the array is made.

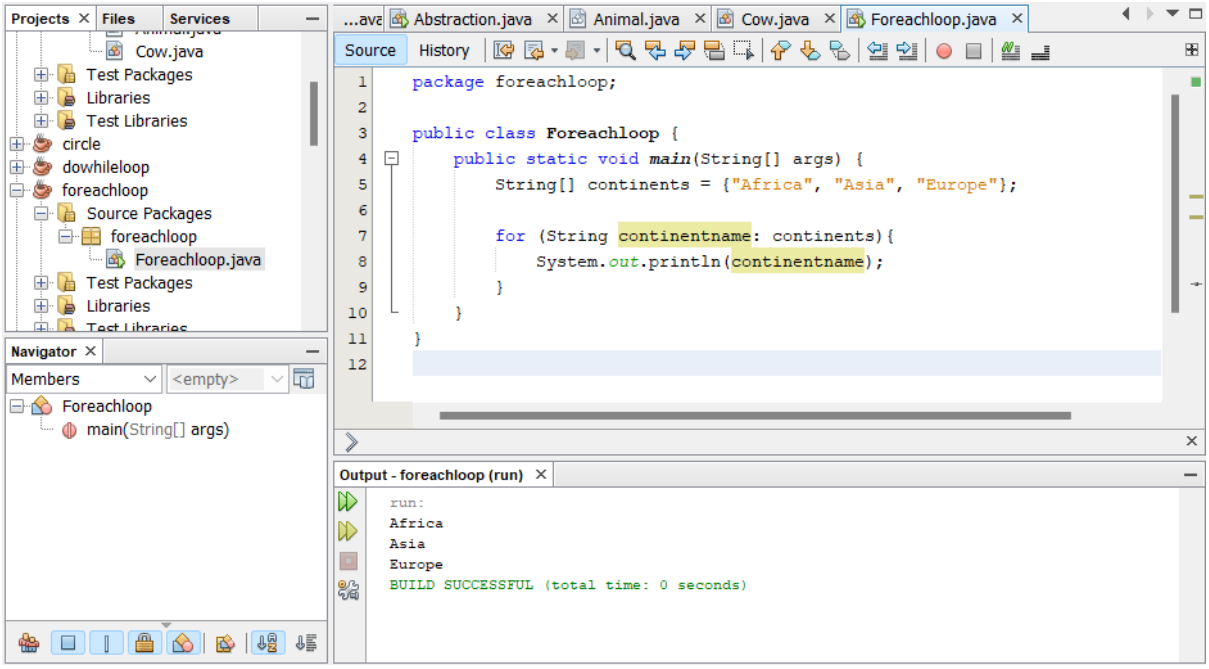


*Figure 2. For-each Loop for Access and Display of Array Items*

The output as shown in the output window of Figure 4 is a clear indicator that the for-each loop was able to iterate through the array elements. The continents variable in the loop acted as a reference to the array while the continentname variable was used to hold array element names each time the loop was iteratively executed.

The for loop, while loop or do … while loop could be used to access and display array elements and their indices in a manner close to the approach used in the for-each loop. For instance, Figure 3 demonstrates an example of how a while loop can be used to access and display array elements with their indices. The output window shows all array items with their indices displayed.

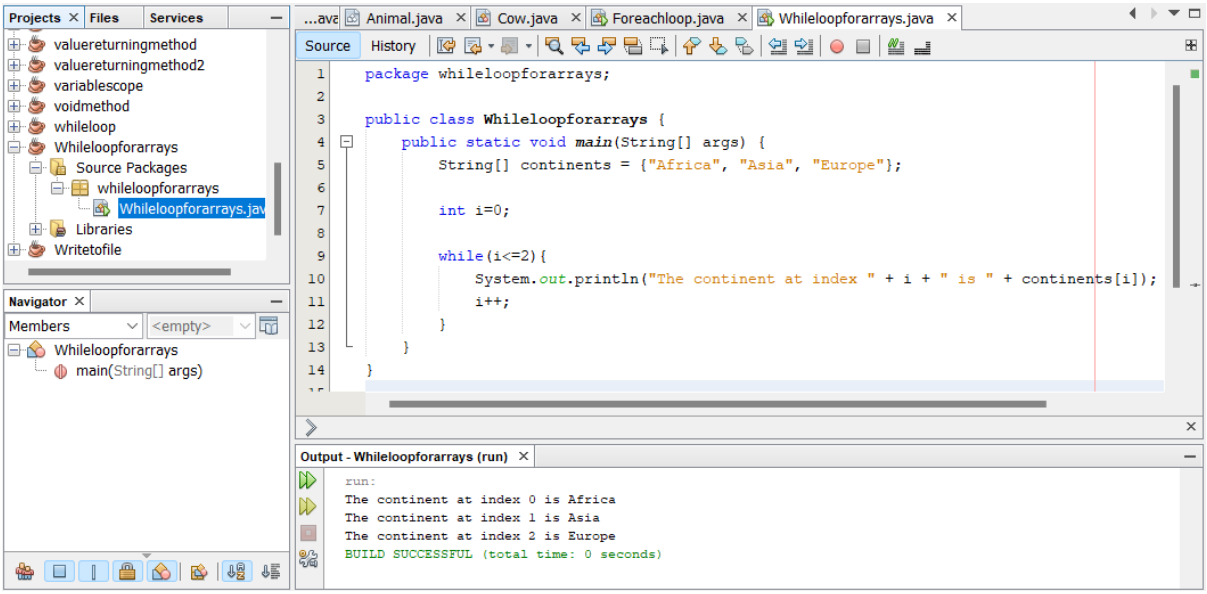


*Figure 3. While Loop for Access and Display of Array Items*

**Summary**

The topic has explored arrays as applied in computer programming. Arrays extend the concept of variables covered in earlier topics. They facilitate for creation of storage location for item sets using a single variable name that acts as the array name. Array items are normally stored the same distance apart in memory and are of the same data type. Any loop can be used to access array items. However, the for-each loop is best suited and specially designed for use with arrays compared to the for loop, while loop, and the do … while loops. The topic has demonstrated how the loops can be used in this context. This topic marks the end of a series of introductory concepts to Java programming. The next set of topics will focus on object-oriented concepts that form the basis for this course.

**Check Points**

1. Narrate your understanding of arrays in comparison to normal variables.
2. Describe the concept of two-dimensional arrays that have not been covered in this topic.
3. Using the for-each loop, demonstrate how all array items can be retrieved and displayed on the console.
4. Using the for loop or do … while loop, demonstrate how all array items can be retrieved and displayed on the console.
5. Using an example of your choice, demonstrate how array items can be accessed and used in operations like arithmetic operations.

**Core Textbooks**

1. Joyce Farrell, Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.
2. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

**Other Resources**

3. Daniel Liang, Y. "Introduction to Java Programming, Comprehensive." (2011).
4. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 4th Edition, Cengage Learning, 2011.
5. Shelly, Gary B., et al. Java programming: comprehensive concepts and techniques. Cengage Learning, 2012.

**References**

[1] Farrell, J., Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.

[2] Malik, D. S., JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

[3] Sebester, R. W., Concepts of Programming Languages, 12th Edition, Pearson, 2018, ISBN 0-321-49362-1.