CORE JAVA - ARRAY

Lecture Notes

A COMPLETE GUIDE FOR BEGINNERS

Softech Solutions Inc.

www.technosolution.tech

saney.alam@techsolutions.tech

Core Java - Array

Introduction

So far in previous chapters we used the variables that hold only one value. An integer variable is capable of storing only one number; likewise a String variable is capable of storing only one string. An array is a type of variable that can store multiple values. It is like a list of items but it always contains similar data type values.
Points to notice about arrays in Java:

Array is a data structure in java that can hold one or more values in a single variable.
Array in java is a collection of similar type of values.
Java has two types of arrays – single dimensional and multidimensional arrays.
Array index starts at 0.

Just to help you to visualize the structure of array, think of an excel sheet column. The data held in a single-list array look like this:

Position	Intereger Array
0	10
1	20
2	30
3	40
4	50
5	60

Position	String Array
0	Ten
1	Twenty
2	Thirty
3	Fourty
4	Fifty
5	Sixty

Each ***cell*** in excel sheet is like a ***compartment*** in an array. Each compartment has its position number. In the image above of ***Integer Array***, position 0 is holding a value of 10, array position 1 is holding a value of 20, and position 2 has a value of 30, and so on. The structure is same for ***String Array*** as well, Ten is stored at the position zero, twenty is stored at one and so on.

Where to use Arrays

Let's say in your Java program you are required to store ten different makes of the car. A normal way of doing this is to create ten different ***string variables*** and store car makes in each one of these variables. It will look like this in normal way of using variable:

```
package javaTutorials;

      public class Array {

      String  sMake1,  sMake2,  sMake3,  sMake4,  sMake5,  sMake6,
      sMake7, sMake8, sMake9,sMake10;
      }
```

Array provides a smart way of doing the trick by storing all the values within just one variable.

Syntax :

```
ArrayDataType[] ArrayName;
```

```
package javaTutorials;
      public class Array {
            String[] sMake;
            }
}
```

Assigning Values to an Array

You can create an array by using the new operator. Declaring an array variable, creating an array, and assigning the reference of the array to the variable can be combined in one statement.

Syntax:

```
ArrayType [] ArrayName = New ArrayType [Size of an Array];
```

ArrayType: This is the type of the array. An array can store different data types like integer, string, boolean etc. But the only condition with it is that all

the values should be of same type in an array. If you try to assign string value to an integer array, you would get compile time error.

ArrayName: Array name can be any name you like to choose. Try making it more logical & meaningful. I always follow a pattern in my code for naming variable. Like for string variables I name my variable starting with small 's', for integer variable I start it with small 'i' and for array I start it with small 'a'. For example: String sExample; int iExample; String[] aExample; Note: This may not be the best practice as 'i' denotes for interfaces in Java.

New: New is the Java keyword to create an object of a class. It locates a block of memory large enough to contain the array.

[Size of an Array]: This decides how big the array is. At the time of creation, the length of the array must be specified and remains constant. [5] denotes that the array have 5 compartments in it. The first index value in the array is zero, thus, the index with value four is used to access the fifth element in the array.

Example – Style 1

```
String[] aCarMake = new String[5];
      aCarMake[0] = "BMW";
      aCarMake[1] = "AUDI";
      aCarMake[2] = "TOYOTA";
      aCarMake[3] = "SUZUKI";
      aCarMake[4] = "HONDA";
```

Arrays can also be created like this:

Example – Style 2

```
String [] aCarMake = {"BMW","AUDI","TOYOTA","SUZUKI","HONDA"};
```

Accessing Array Values

A program can access each of the array elements (the individual cells) by referring to the name of the array followed by the subscript denoting the element (cell). For example, the third element is denoted ***aCarMake[2].*** Array also gives a property called ***length***. Length returns the size of the array.

Example

This will produce the following result –

```
package javaTutorials;

public class Array {

      public static void main(String[] args) {
            String [] aMake =
{"BMW","AUDI","TOYOTA","SUZUKI","HONDA"};

            //This is to store the size of the Array
            int iLength = aMake.length;
            System.out.println("Length of the Array is ==> " +
iLength);

            //This is to access the first element of an array
directly with it's position
            String sBMW = aMake[0];
            System.out.println("First value of the Array is ==> " +
sBMW);

            //This is to access the last element of an Array
            String sHonda = aMake[iLength-1];
            System.out.println("Last value of the Array is ==> " +
sHonda);

            //This is to print all the element values of an Array
            for(int i = 0;i<=iLength-1;i++){
                  System.out.println("The value stored at position
"+i+" in aMake array is ==> " + aMake[i]);
            }
      }

}
```

Output

Length of the Array is ==> 5
First value of the Array is ==> BMW
Last value of the Array is ==> HONDA
The value stored at position 0 in aMake array is ==> BMW
The value stored at position 1 in aMake array is ==> AUDI
The value stored at position 2 in aMake array is ==> TOYOTA

The value stored at position 3 in aMake array is ==> SUZUKI
The value stored at position 4 in aMake array is ==> HONDA

Passing an Array to a Method

Array can be easily passed to a method as parameters, just like we pass integers and strings to methods. In the below example aMake array is passed to Print_Array() method. Here is an example:

Example

```
package javaTutorials;
public class Passing_Array {
	public static void main(String[] args) {
		//Declaring an Array
		String [] aMake = {"BMW","AUDI","TOYOTA","SUZUKI","HONDA"};
		// Calling Print Array method and passing an Array as a parameter
		Print_Array(aMake);

	}

	//This accept Array as an argument of type String
	public static void Print_Array(String []array){

		for(int i = 0;i<=array.length-1;i++){
			System.out.println("Printing all the values of an Array ==> " + array[i]);
		}
	}

}
```

Output

Printing all the values of an Array ==> BMW
Printing all the values of an Array ==> AUDI
Printing all the values of an Array ==> TOYOTA
Printing all the values of an Array ==> SUZUKI
Printing all the values of an Array ==> HONDA

Returning an Array from a Method

A method may also return an array. The way we return any other data type from a method, array can also be returned in the same way. Just mention the type of array in the method ***declaration*** followed by **[ ].** In the below example an array of type ***string*** is returned from a method ***ReturnArray().***

Example

```
package javaTutorials;
public class Returning_Array {
        public static void main(String[] args) {
                String[] aMake = ReturnArray();
                for(int i = 0;i<=aMake.length-1;i++){
                        System.out.println("Printing all the values of an Array ==> " + aMake[i]);
                        }
                }

        //This method returns an Array of type String
        public static String[] ReturnArray() {
                String [] aArray = {"BMW","AUDI","TOYOTA","SUZUKI","HONDA"};
                  return aArray;
                }
        }
```

This will produce the following result –

Output

```
Printing all the values of an Array ==> BMW
Printing all the values of an Array ==> AUDI
Printing all the values of an Array ==> TOYOTA
Printing all the values of an Array ==> SUZUKI
Printing all the values of an Array ==> HONDA
```