

# **Core Java 8 and Development Tools**

Lesson 10 : Arrays

## Lesson Objectives

- After completing this lesson, participants will be able to
  - Understand the different types of Arrays
  - Implement one and multi dimensional arrays
  - Iterate arrays using loops
  - Use varargs
  - Work with `java.util.Arrays`



Copyright © Capgemini 2015. All Rights Reserved 2

This lesson discusses about how to work java arrays.

Lesson outline:

- 10.1: One dimensional array
- 10.2: Multidimensional array
- 10.3: Using varargs
- 10.4: Using Arrays class
- 10.5: Best Practices

10.1: One dimensional array

## Arrays

- Arrays are used to group elements of either of primitive or reference types
- Array in java is created as Object:
  - This object will help developers to find size of array
  - Using this object developers can manipulate array
  - Can be compared with null
- All elements of array of same type
- Array is a fixed-length data structure having zero-based indexing



Copyright © Capgemini 2015. All Rights Reserved 3

Arrays can be created for either primitive or reference type elements. Array in java is created as Object using new operator. Once array is created, individual elements can be accessed using index number enclosed in square brackets.

Arrays indexing is zero based, it means the first element of array start at index 0, second element is at 1, and so on. The last element of array is indexed as one minus size of an array.

Array size can be captured by using public final instance variable called length. This feature will avoid any runtime exceptions resulted due to out of bounds access.

10.1: One dimensional array

## Arrays

- A group of like-typed variables referred by a common name
- Array declaration and initialization:
  - `int arr [];`  
`arr = new int[10];`
  - `int arr[] = {2,3,4,5};`
  - `int twoDim [][] = new int[4][5];`



Copyright © Capgemini 2015. All Rights Reserved

4

Syntax wherein the array is declared and initialized in the same statement:

```
strWords = { "quiet", "success", "joy", "sorrow", "java" };
```

10.1: One dimensional array

## Creating Array Objects

- Arrays of objects too can be created:

- Example 1:

```
Box barr[] = new Box[3];  
barr[0] = new Box();  
barr[1] = new Box();  
barr[2] = new Box();
```

- Example 2:

```
String[] Words = new String[2];  
Words[0]=new String("Bombay");  
Words[1]=new String("Pune");
```



Copyright © Capgemini 2015. All Rights Reserved

5

Use new operator or directly initialize an array. When you create an array object using new, all its slots are initialized for you (0 for numeric arrays, false for boolean, '\0' for character arrays, and null for objects).

Like single dimensional arrays, we can form multidimensional arrays as well.

Multidimensional arrays are considered as array of arrays in java and hence can have asymmetrical arrays. See an example next.

10.1: One dimensional array

## Demo

- Executing the ArrayDemo.java program



Copyright © Capgemini 2015. All Rights Reserved 6

We have seen how to pass parameters to program during compile time using parameter passing. One can pass parameters to a program at runtime too. The args parameter (a String array) in main() receives command line arguments.

```
class ArrayDemo {
    int intNumbers[];
    ArrayDemo(int i) {
        intNumbers = new int[i];
    }
    void populateArray() {
        for(int i = 0; i < intNumbers.length; ++i) intNumbers[i] = i;
    }
    void displayContents() {
        for(int i = 0; i < intNumbers.length; ++i)
            System.out.println("Number " + i + ": " + intNumbers[i]);
    }
    public static void main(String[] args) {
        //Accepting array length as command line argument.
        int intArg = Integer.parseInt(args[0]);
        ArrayDemo ad = new ArrayDemo(intArg);
        ad.displayContents();
        ad.populateArray();
        ad.displayContents();
    }
}
```

10.1: One dimensional array

## Enhanced for Loop (foreach)

- New feature introduced in Java 5
- Iterate through a collection or array

- Syntax:

```
for (variable : collection)
{ //code }
```

- Example

```
int sum(int[] intArray)
{
    int result = 0;
    for (int index : intArray)
        result += index;
    return result;
}
```



Copyright © Capgemini 2015. All Rights Reserved

7

Enhanced for loop works with collections and arrays. Notice the difference between the old code where the three standard steps of initialization, conditional check and re-initialization are explicitly required to be mentioned.

Old code

```
public class OldForArray {
    public static void main(String[] args){
        int[] squares = {0,1,4,9,16,25};
        for (int i=0; i< squares.length; i++){
            System.out.printf("%d squared is %d.\n",i, squares[i]);
        }
    }
}
```

New Code

```
public class NewForArray {
    public static void main(String[] args) {
        int j = 0;
        int[] squares = {0, 1, 4, 9, 16, 25};
        for (int i : squares) {
            System.out.printf("%d squared is %d.\n", j++, i);
        }
    }
}
```

10.2: Multi dimensional array

## Arrays

- 2 Dimensional Array declaration and initialization:

- `int arr[][] = { {2,3},{4,5},{34,56}};`
- `int twoDim [][] = new int[4][5];`



Copyright © Capgemini 2015. All Rights Reserved 8

Syntax wherein the array is declared and initialized in the same statement:

```
strWords = { "quiet", "success", "joy", "sorrow", "java" };
```



10.3 : Using Varargs

## Variable Argument List

- New feature added in J2SE5.0
- Allows methods to receive unspecified number of arguments
- An argument type followed by ellipsis(...) indicates variable number of arguments of a particular type
  - *Variable-length* argument can take from zero to *n* arguments
  - **Ellipsis** can be used only once in the parameter list
  - **Ellipsis** must be placed at the end of the parameter list



Copyright © Capgemini 2015. All Rights Reserved 9

J2SE5 has added a new feature that simplifies the creation of methods that need to take a variable number of arguments. This feature is called **varargs** and it is short for variable-length arguments.

10.3 : Using Varargs

## Variable Argument List (contd..)

- The above print function can be invoked using any of the invocations:

- `print(1,1,"XYZ")`
- `print(2,5)`
- `print(5,6,"A","B")`

```
//Valid Code
void print(int a,int b,String...c)
{
    //code
}
```

```
//Invalid Code
void print(int a, int b...,float c)
{
    //code
}
```

Varargs can be used only in the final argument position.




Copyright © Capgemini 2015. All Rights Reserved 10


Note that the ellipses (...) must come only after the last parameter. Putting it anywhere else is invalid.

The three periods indicate that the final argument may be passed as an array or as a sequence of arguments.

10.3 : Using Varargs  
**Demo**

- Lesson-10-Execute the varargs.java program



 Copyright © Capgemini 2015. All Rights Reserved 11

```
import static java.lang.System.*;
public class varargs {
    static void print(int a,int y,String...s) {
        out.println(a+" "+y);
        for(int i=0;i<s.length;i++) out.print(s[i]+" ");
        out.println();
    }
    public static void main(String[] arg) {
        print(3,2,"java","java 5");
        out.println("Next invoke");
        print(1,2,"a","b","c","d","e");
    } }
```

Java Arrays expose a property called length that returns the length of the array.

O/P:

32

java java 5

Next invoke

12

a b c d e

10.4: Using Arrays Class

## Using java.util.Arrays Class

- This class contains lots of useful methods to manipulate contents of array

Method Name	Use
asList	Creates a new List from array
binarySearch	Use to search an element in an array
copyOf(array,n)	Creates new array of n size and copy all elements from array to new one
copyOfRange(array,n,from,to)	Creates new array of n size and copy specified elements from array to new one
sort	Sort elements of an array
equals	Compare two array elements
fill	Inserts specified value to each element of an array
stream(array)	Creates stream from an array



Copyright © Capgemini 2015. All Rights Reserved 12

### Arrays Class:

Java provides utility Arrays class to manipulate arrays. This class is provided in the java.util package and includes lots of static methods to deal with array.

Please refer the java documentation of this class to know more about method signature and use.

Note: stream() method will be discussed in later chapter.

10.4: Using Arrays Class

## Demo

- Execute the NewForArray.java program



Copyright © Capgemini 2015. All Rights Reserved 13

```
import java.util.Arrays;
public class UsingArrays{
    static void sort(int...s) {
        Arrays.sort(s);
        for(int i=0;i<s.length;i++) out.print(s[i]+" ");
        out.println();
    }
    public static void main(String[] arg) {
        sort(15,20,42,3,56,34);
    } }
```

## Summary

- In this lesson, you have learnt about:

- Creating and using array
- Manipulating array
- Iterating array
- Varargs
- Using java.util.Arrays class



## Review Question

- Question 1: If a display method accepts an integer array and returns nothing , is following call to display method is correct? State true or false.
  - `display( {10,20,30,40,50})`
- Question 2: All methods in `java.util.Arrays` class are static (excluding `Object` class methods).
  - True/False

