



SNS COLLEGE OF TECHNOLOGY



Coimbatore-35.

An Autonomous Institution

**Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A++’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
COURSE CODE & NAME : 23CST205 - Object Oriented Programming Using Java

II YEAR/ III SEMESTER

UNIT – II INTRODUCTION TO JAVA

Topic: BASICS OF JAVA PROGRAMMING-ARRAYS IN JAVA



Java Arrays



Java Arrays

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To declare an array, define the variable type with square brackets:

String[] cars;

- We have now declared a variable that holds an array of strings. To insert values to it, you can place the values in a comma-separated list, inside curly braces:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

- To create an array of integers, you could write:

int[] myNum = {10, 20, 30, 40};



Java Arrays



Access the Elements of an Array

- You can access an array element by referring to the index number.
- This statement accesses the value of the first element in cars:
- Example:

```
public class Main {  
    public static void main(String[] args) {  
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
        System.out.println(cars[0]);  
    }  
}
```

- **Output:** Volvo
- **Note:** Array indexes start with 0: [0] is the first element. [1] is the second element, etc.



Java Arrays



Change an Array Element

- To change the value of a specific element, refer to the index number:
- Example:

```
cars[0] = "Opel";
```

```
public class Main {  
    public static void main(String[] args) {  
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
        cars[0] = "Opel";  
        System.out.println(cars[0]);  
    }  
}
```

- **Output:** Opel



Java Arrays



Array Length

- To find out how many elements an array has, use the length property:
- Example

```
public class Main {  
    public static void main(String[] args) {  
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
        System.out.println(cars.length);  
    }  
}
```

- Output: 4



Java Arrays Loop

Loop Through an Array

- You can loop through the array elements with the for loop, and use the length property to specify how many times the loop should run.
- The following example outputs all elements in the cars array:
- Example

```
public class Main {  
    public static void main(String[] args) {  
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
        for (int i = 0; i < cars.length; i++) {  
            System.out.println(cars[i]);  
        }  
    }  
}
```

Output:

Volvo
BMW
Ford
Mazda



Java Arrays Loop



Loop Through an Array with For-Each

- There is also a "for-each" loop, which is used exclusively to loop through elements in arrays:
- Syntax

```
for (type variable : arrayname)
{
    ...
}
```



Java Arrays Loop



Loop Through an Array with For-Each

- The following example outputs all elements in the cars array, using a "for-each" loop:
- Example

```
public class Main {  
    public static void main(String[] args) {  
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
        for (String i : cars) {  
            System.out.println(i);  
        }  
    }  
}
```

Output:

Volvo
BMW
Ford
Mazda

- The example above can be read like this: for each String element (called i - as in index) in cars, print out the value of i.
- If you compare the for loop and for-each loop, you will see that the for-each method is easier to write, it does not require a counter (using the length property), and it is more readable.



Java Multi-Dimensional Arrays



Multidimensional Arrays

- A multidimensional array is an array of arrays.
- Multidimensional arrays are useful when you want to store data as a tabular form, like a table with rows and columns.
- To create a two-dimensional array, add each array within its own set of curly braces:
- Example:

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
```

- myNumbers is now an array with two arrays as its elements.



Java Multi-Dimensional Arrays



Access Elements

- To access the elements of the myNumbers array, specify two indexes: one for the array, and one for the element inside that array.
- This example accesses the third element (2) in the second array (1) of myNumbers:

Example:

```
public class Main {  
    public static void main(String[] args) {  
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };  
        System.out.println(myNumbers[1][2]);  
    }  
}
```

Output:7

- **Remember that:** Array indexes start with 0: [0] is the first element. [1] is the second element, etc.



Java Multi-Dimensional Arrays



Change Element Values

- You can also change the value of an element:
- Example:

```
public class Main {  
    public static void main(String[] args) {  
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };  
        myNumbers[1][2] = 9;  
        System.out.println(myNumbers[1][2]); // Outputs 9 instead of 7  
    }  
}
```

Output:9



Java Multi-Dimensional Arrays



Loop Through a Multi-Dimensional Array

- You can also use a for loop inside another for loop to get the elements of a two-dimensional array (we still have to point to the two indexes):
- Example:

```
public class Main {  
    public static void main(String[] args) {  
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };  
        for (int i = 0; i < myNumbers.length; ++i) {  
            for(int j = 0; j < myNumbers[i].length; ++j) {  
                System.out.println(myNumbers[i][j]);  
            }  
        }  
    }  
}
```

Output:

1
2
3
4
5
6
7



Java Multi-Dimensional Arrays



Loop Through a Multi-Dimensional Array

- You could just use a for-each loop, which is considered easier to read and write:
- Example:

```
public class Main {  
    public static void main(String[] args) {  
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };  
        for (int[] row : myNumbers) {  
            for (int i : row) {  
                System.out.println(i);  
            }  
        }  
    }  
}
```

Output:

1
2
3
4
5
6
7

