Lesson:

Arrays in Java

Pre-Requisites

- Basic Java syntax
- Java methods

List of Concepts Involved

- Array Introduction
- Declaration
- Creation
- Array Types
- Operations
- Problems based on arrays

Suppose we have to write a program in which can accept salaries of 50 employees. If we solve this problem by making use of variables, we need 50 variables to store employees' salaries. Managing these 50 variables is not an easy task and will make the program a complex and lengthy one. This problem can be solved by declaring 1 array having 50 elements. Did you notice how convenient the scenario became? If arrays are that useful, why not learn about them?

Topic: Array Introduction

Array is a data structure (storage, used to store and organize data) to store a group or collection of (homogenous data) items, sequentially, inside memory. Homogenous data is data of the same type, for example, integer or string or floating number, etc. Each array element is identified with an index number.

- The indexing of an array is 0-based i.e. the first element is at index 0, second element is at index 1 and so on. The desired array element can be directly and individually accessed using these numbered indexes.
- The memory allocation in arrays is contiguous i.e. elements are stored one after another.
- An array can be single-dimensional or multi-dimensional based on the utility and application.

Topic: Array Declaration and Creation

- An array can be created using a new keyword in Java.

```
type var-name[];
     OR
type[] var-name;
```

Examples:
Primitive

```
int intArray[];
or int[] intArray;

byte byteArray[];
short shortsArray[];
boolean booleanArray[];
long longArray[];
float floatArray[];
double doubleArray[];
char charArray[];
```

Object

```
MyClass myClassArray[];
Object[] ao;
Collection[] ca;
```

- **Array initialization**
  datatype [] var-name = new type [size];
  Example: `int[] intArray = new int[20];`

- **Array Literal**
  With curly braces, we can initialize the array and add value to it during initialization without defining the size.
  `int[] intArray = { 1,2,3,4,5,6,7,8,9,10 };`

You might be wondering that why arrays should be used when we already have the provision of creating as many variables as we want/need.
Look at the scenarios mentioned below which will clear the air around utility of arrays for our good.

Case 1: Scenario without Array
In the example below, we are using five different variables to save our elements one by one.

```
public class Color
{
  public static void main(String[] args) {
    String colour1 = "Red";
    String colour2 = "Green";
    String colour3 = "Blue";
    String colour4 = "Yellow";
    String colour5 = "Purple";
    // To print all the elements to the console
    System.out.println(colour1);// Red
    System.out.println(colour2);// Green
    System.out.println(colour3);// Blue
    System.out.println(colour4);// Yellow
    System.out.println(colour5);// Purple
}
}
```

Woah ! It was cumbersome and may result in manual mistakes. Isn’t it ?

Case 2: Using the array concept
Now, we will implement the same scenario using an array.

```
public class Color
{
  public static void main(String[] args) {
    // To create an array of colours to store values
    String colours[] = {"Red", "Green", "Blue", "Yellow", "Purple"};
    // To print all the elements entered in a array to the console
    for (int i = 0; i < 5; i++) {
      System.out.println(colours [i]);
}
}
}
```

Output:
Red
Green
Blue
Yellow
Purple

Thats it ?! Yes, absolutely !
Did you see how arrays made the implementation so convenient and saved a lot of our coding time !

Let us understand how these approaches are different:

- Without an array, each value is stored in different variables, which results in random memory allocation for each variable; whereas when we used the concept of array, the values were stored in contiguous memory locations. Hence, handling and accessing data became extremely convenient.

Arrays have an unlimited potential if we use them correctly. Let us look at its types and see how versatile these can be to handle the most complex scenarios (which will be discussed in the forthcoming lectures)

Topic: Array Types

There can be many classifications in arrays but we will concentrate on the most widely used ones that are based on the dimension of data that it is handling.

1.Single-dimensional Array:
When we have elements stored in a single dimension sequentially, they are called single-dimensional arrays. We can declare and allocate memory to a single-dimensional array using a single variable in Java.
Here is an example,

```
public class Color
{
  public static void main(String[] args) {
    String[] colours = {"red", "green", "blue"};
    // To print the elements in the console:
    for(String colour : colours)
      System.out.print(colour + ", " );
}
}
```

Output:
red, green, blue

Syntax for declaring a single dimension array -

```
int[] <name_of_array> = {element_0, element_1, element_2, element_3, ...
elementN};
```

Different approach to create array in Java:

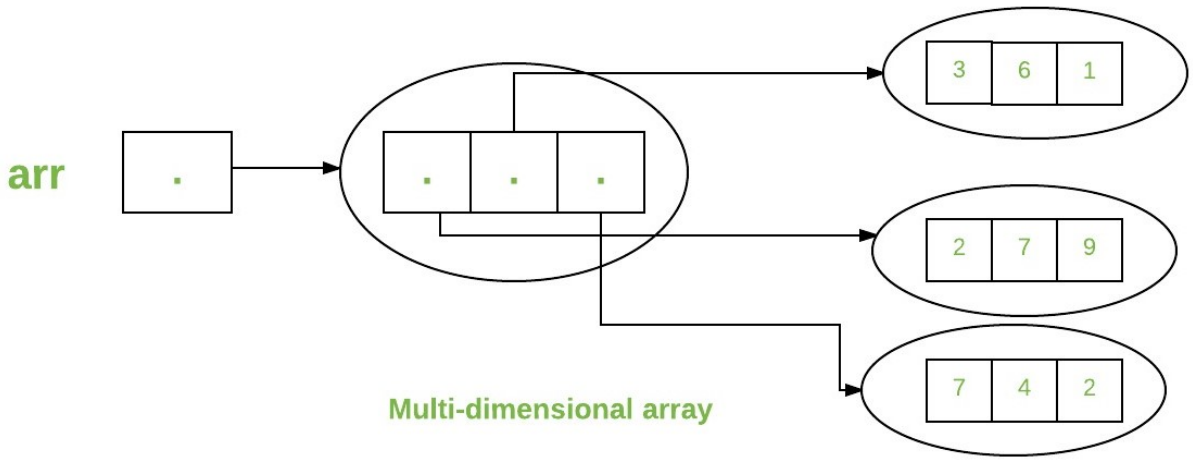
```
public class Color
{
  public static void main(String[] args) {
  // To create an Single-Dimension array:
  int[] myArray = { 1, 2, 3, 4 , 5 };
  // first element
  System.out.println(myArray[0]); // to print 1
  // second element
  System.out.println(myArray[1]); // to print 2
  // Third element
  System.out.println(myArray[2]); // to print 3
  // fourth element
  System.out.println(myArray[3]); // to print 4
  // fifth element
  System.out.println (myArray[4]); // to print 5
}
}
```

Output:

1
2
3
4
5

Multi-dimensional Array:

A multidimensional array is simply an array that consists of two or more dimensions and is also commonly referred to as an array of arrays.
The diagram below will help you visualize the actual implementation structure of a multi- dimensional array.



To build a two-dimensional array, wrap each array in its pair of "[]" square brackets.
Look at the example below for clarity. Here we are creating a 2-dimensional array

```
int[][] items = {
{2, 3},
{5, 6},
{7, 8}
};
```

Topic : Operations in Arrays

Here, we are going to discuss the most basic operations that can be done on a

1. Length of array

```
public class Color
{
  public static void main(String[] args) {
    String[] colours = {"Red", "Green"," Yellow"," Purppe"};
    int length = colours.length;
    System.out.println(length); // print 4
}
}
```

Important Note: Array index always starts from 0, which means the first element is stored at index 0 and the last element will be stored at index length-1.

2. Looping through Array

There are many ways to iterate over the Array. The most common ways of looping through arrays in Java are:

- **For Loop**

```
public class Color
{
  public static void main(String[] args) {
    // Creating an Array
    String[] colours = {"Red", "Green"," Yellow"," Purple"};
    // Using loop to iterate over array
    int length = colours.length;
    // Printing array elements using index
    System.out.println(length); // print 4
}
}
```

- **For each loop**

This loop helps in iterating over iterable objects like String, Array and so on.

Syntax:

```
for (<data_type> <variable_name> : <array_name> ) {
statement;
}
```

Limitation:

- Its limitation is that it can only be used for traversing the whole array and not part of the array.

Example:

```
public class Color
{
  public static void main(String[] args) {
    int[] a = { 1, 2, 3, 4, 5, 6, 7, 8 };
    // iterating over an array
    for (int i : a) {
      // accessing each element of array
      System.out.println(i);
}
}
}
```

Output:

1
2
3
4
5
6
7
8

Topic: Arrays Problems

Problem 1: Calculate the sum of all the elements in the given array.
Input : arr[] = {1, 5, 3}
Output: 9

Code:

```
public class Color
{
  public static void main(String[] args) {
    int[] arr = { 1, 5, 3};
    int sum = 0;
    for(int i = 0; i < arr.length; i++) {
      sum += arr[i];
    }
  System.out.println(sum);
}
}
```

Problem 2: Calculate the maximum value out of all the elements in the array.
Input : arr[] = {1, 5, 3}
Output : 5

Code:

```
public class Color
{
  public static void main(String[] args) {
    int[] arr = { 1, 5, 3};
    int mx = arr[0];
    for(int i = 0; i < arr.length; i++) {
      if(arr[i] > mx) mx = arr[i];
    }
    System.out.println(mx);
}
}
```

Problem 3: Search if the given element x is present in the array or not and find the index. If not present then return the index as -1. (Linear Search)
Input : arr[] = {1, 5, 3}
x = 5
Output : 1

Code:

```
public class Color
{
  public static void main(String[] args) {
  int[] arr = { 1, 5, 3};
  int x=5;
  int index = -1;
  for(int i = 0; i < arr.length; i++) {
    if(arr[i] == x) index = i;
  }
  System.out.println(index);
}
}
```

That is all for this class ! See you in the next array lecture !! Keep learning ! Keep Exploring !!

Upcoming Class Teaser :

- Advance Array Concepts