Lesson Plan

Java Array-1

PW SKILLS

Pre-requisites:

- Java Syntax
- Loops

List of patterns involved:

1. Introduction to Arrays
2. Syntax, accessing elements of Arrays
3. Printing Output and Taking Input
4. Types of Arrays
5. Length operator
6. Memory Allocation of Arrays in Java
7. Linear search
8. Basic problems

What is an array?

- Arrays can be described as a type of data structure used to store a group or collection of items or elements sequentially inside a memory.
- The main function of an Array is to store a collection of homogenous data of the same type. For example, integers, strings, floating numbers, etc.
- The indexing of an Array is 0 based indexing ie the first element is at index 0 and the second element is at index 1 and so forth. We can directly access the required elements using the indexes.
- The memory allocation in Arrays is contiguous in nature.
- We can create both single-dimensional and multidimensional arrays.

Syntax and Declaration

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

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

OR

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

- Some 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**

```
var-name = new type [size];
```

Example:

```
int[] intArray = new int[20];
```

- **Array Literal**

  With curly braces we can initialise the array and add value to it during initialization without defining the size.

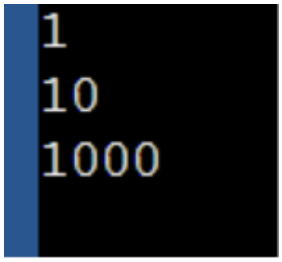
```
int[] intArray = { 1,2,3,4,5,6,7,8,9,10 };
```

How to access Element in Array?

Elements in array can be accessed using [ ] brackets.

```
int[] myArray = { 1,2,3,4,5,6,7,8,9,10 };
System.out.println(myArray[0]);
System.out.println(myArray[9]);
myArray[0] = 1000;
System.out.println(myArray[0]);
```

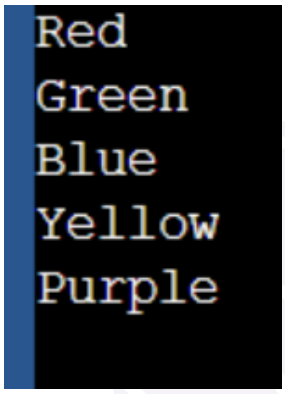
Output:



Example:

```
// To create an array of colors to store values
String colors[] = {“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:



Printing Output and Taking Input

In Java, you can use the System.out.print or System.out.println methods to print output to the console, and the Scanner class to take input from the user.

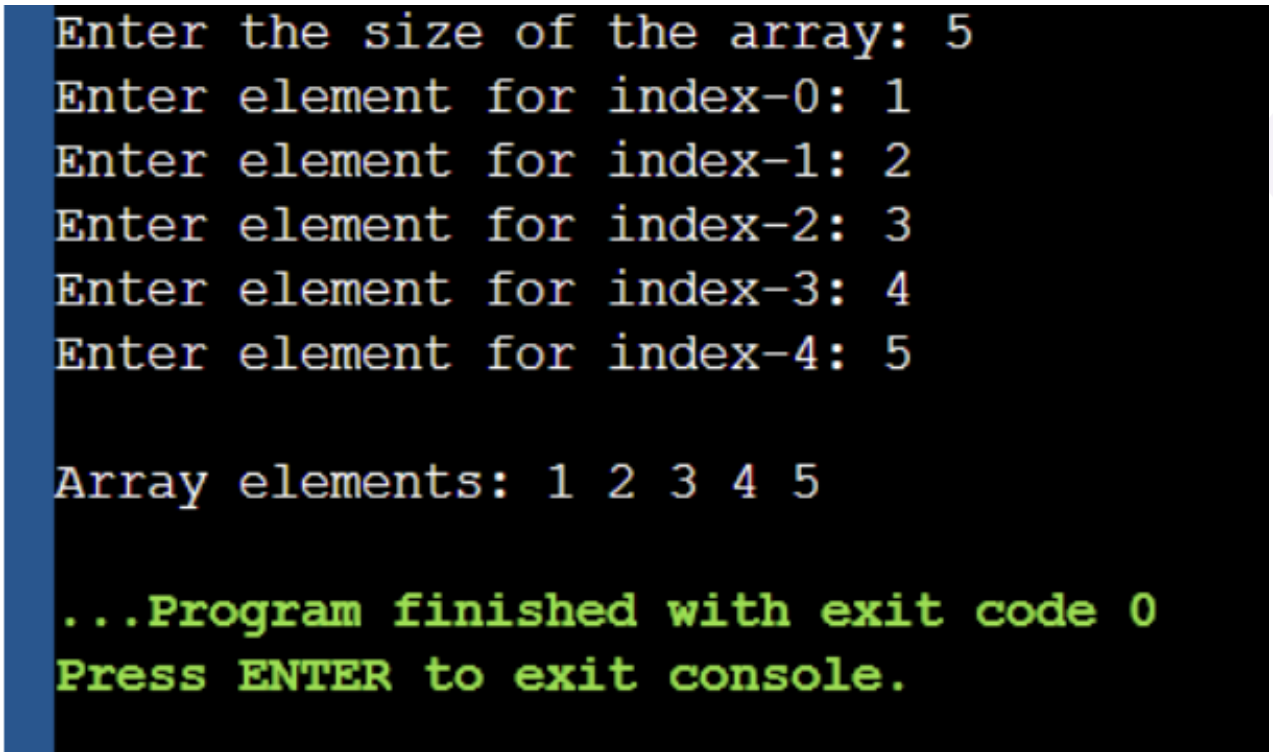
Example:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int size = scanner.nextInt();
        int[] userArray = new int[size];
        for (int i = 0; i < size; i++) {
```

```
            System.out.print("Enter element for index-" + i + ":
");
            userArray[i] = scanner.nextInt();
        }
        System.out.print("\nArray elements: ");
        for (int i = 0; i < size; i++) {
            System.out.print(userArray[i] + " ");
        }
        scanner.close();
    }
}
```

Output:



Q. Given an array of marks of students, if the mark of any student is less than 35 print its roll number. [roll number here refers to the index of the array.]

```
public class Main {
    public static void main(String[] args) {
        int[] marks = {28, 77, 45, 60, 90, 32, 55};

        System.out.println("Roll numbers of students with marks
less than 35 are:\n");
        for (int i = 0; i < marks.length; i++) {
            if (marks[i] < 35) {
```

```
                System.out.println("Roll Number: " + i);
            }
        }
    }
}
```

```
Roll numbers of students with marks less than 35 are:

Roll Number: 0
Roll Number: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Q. Are the following array declarations correct?

1. **int a (25) ;**
2. **int size = 10, b[size] ;**
3. **int c = {0,1,2} ;**

Ans.

A. This is not a correct way to declare an array in Java. The correct syntax for declaring an array would be something like int[] a = new int[25]; if you want an array of size 25.

B. This is not a valid way to declare an array in Java. In Java, the size of an array must be known at compile-time. If you want an array of size 10, you should do it like this: int[] b = new int[10];

C. This is also incorrect. To declare and initialize an array in one line, you should use curly braces {} with the new keyword or square brackets. Correct syntax would be: int[] c = {0, 1, 2};

Q. Which element of the array does this expression reference? num[4]

Ans.

The expression num[4] references the element at index 4 of the array num. In Java and many other programming languages, array indices typically start from 0. Therefore, num[4] would refer to the fifth element in the array num.

Example:

```
int[] num = {10, 20, 30, 40, 50};
```

Then, num[4] would be equal to 50 because it's the value stored at the fifth position in the array (remembering that indexing starts from 0).

Array Types

- Single dimensional or one-dimensional array
- Multidimensional Array

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,

```
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 an single dimension array:

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

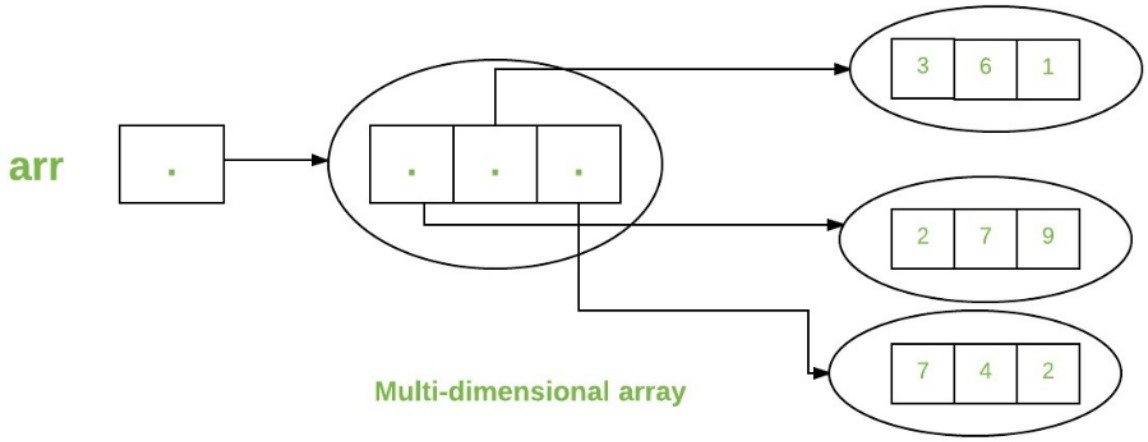
Different approach to create array in Java:

```
// To create an Single-Dimension array:
int[] myArray = { 1, 2, 3, 4 , 5 };
System.out.println(myArray[0]); // to print 1
System.out.println(myArray[1]); // to print 2
System.out.println(myArray[2]); // to print 3
System.out.println(myArray[3]); // to print 4
System.out.println (myArray[4]); // to print 5
```

Output:

1
2
3
4
5

Multi dimensional Array:



- A Multidimensional Array can be described as an array that consists of two or more than two-dimensions.
- A multidimensional array is also termed an Array of arrays.
- To build a two-dimensional array, wrap each Array in its pair of "[]" square brackets.

Example for Multi Dimensional Array :

Example to create multi-dimensional Array

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

Length Operator

```
String[] colours = {"Red", "Green"," Yellow"," Purple"}
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.

Memory Allocation of Arrays in Java

In Java, arrays are objects, and their memory allocation is managed by the Java Virtual Machine (JVM). When you create an array in Java, the JVM allocates memory for it on the heap.

Here are the main steps in the memory allocation of arrays in Java:

1. **Declaration:** When you declare an array in Java, you are essentially creating a reference to an array object. For example:

```
int[] numbers;
```

2. **Instantiation (Creating the Object):** To create an array object and allocate memory for it, you use the new keyword:

```
numbers = new int[5];
```

This line creates an array of integers with a size of 5. The new keyword is used to instantiate the array object, and int[5] specifies the size of the array.

3. **Initialization:** You can initialize the array at the time of declaration or later. For example:

```
int[] numbers = {1, 2, 3, 4, 5};
```

This initializes the array with the specified values.

4. **Accessing Elements:** You can access individual elements of the array using index notation. For example:

```
int x = numbers[2];
```

// Access the third element of the array

5. **Garbage Collection:** Java has an automatic garbage collector that is responsible for reclaiming memory occupied by objects that are no longer in use. When there are no references to an array, it becomes eligible for garbage collection.

The memory for the array is allocated on the heap, which is a region of the computer's memory used for dynamic memory allocation. The JVM manages the allocation and deallocation of memory, providing a level of abstraction that simplifies memory management for the programmer.

Practice Questions

Q1. Predict the output :

```
public class Main {
    public static void main(String[] args) {
        int[] num = new int[26];
        num[0] = 100;
        num[25] = 200;
        int temp = num[25];
        num[25] = num[0];
        num[0] = temp;
        System.out.println("\n" + num[0] + " " + num[25]);
    }
}
```

Ans. 200 100

Let's analyze the code step by step:

1. An array num of size 26 is created.
2. num[0] is assigned the value 100.
3. num[25] is assigned the value 200.
4. The value of num[25] (200) is stored in the temporary variable temp.
5. The value of num[0] is then updated with the value of num[25] (200).
6. The value of num[25] is updated with the value stored in the temporary variable temp (which is 200).
7. Now, when you print num[0] and num[25], it will be 200 100.

Q2. Point out the errors(if any) in the following code:

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int size = scanner.nextInt();
        int[] arr = new int[size];
        for (int i = 0; i < size; i++) {
            arr[i] = scanner.nextInt();
            System.out.print(arr[i] + " ");
        }
    }
}
```

Ans. The code is mostly correct, but it's missing an import statement for the **Scanner class**. To fix this issue, we need to add the following import statement at the beginning of your code:
import java.util.Scanner;

Q3. Calculate the sum of all the elements in the given array.

Input: arr[] = {1, 5, 3}
Output: 9

Code:

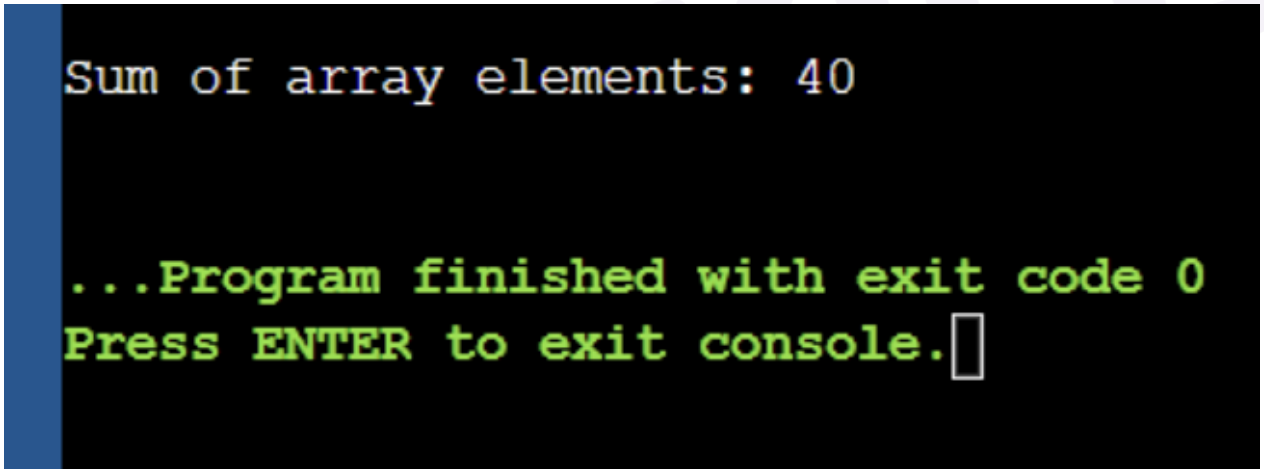
```
public class Main {
    public static void main(String[] args) {

        int[] arr = {2, 5, 8, 10, 15};

        int sum = 0;
        for (int i = 0; i < arr.length; i++)
            sum += arr[i];

        System.out.println("\nSum of array elements: " + sum);
    }
}
```

Output:



Linear Search

Q4. Search if the given element x is present in the array or not and find the index.

Input: arr[] = {1, 5, 3}, x = 5
Output: 1

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {2, 5, 8, 10, 15};
        int x = 8;
        int index = -1;

        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == x) {
                index = i;
                break;
            }
        }

        if (index != -1) {
            System.out.println("Element " + x + " found at index:
" + index);
        } else {
            System.out.println("Element " + x + " not found in
the array.");
        }
    }
}
```

Output:

```
Element 8 found at index: 2


...Program finished with exit code 0
Press ENTER to exit console.
```

```
Element 999 not found in the array.


...Program finished with exit code 0
Press ENTER to exit console.
```

Q5. Calculate the maximum value out of all the elements in the array.

Input: arr[] = {1, 5, 3}
Output: 5

Code:

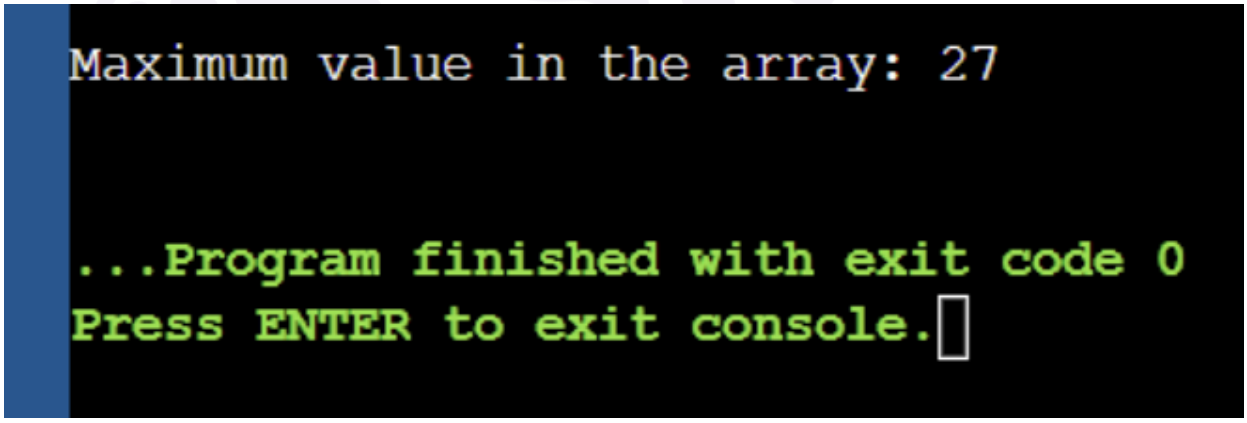
```
public class Main{
    public static void main(String[] args) {
        int[] arr = {12, 5, 27, 8, 10, 15};

        int max = arr[0]; // Assume the first element as the
initial maximum

        for (int i = 1; i < arr.length; i++) {
            if (arr[i] > max) {
                max = arr[i]; // Update max if a larger element
is found
            }
        }

        System.out.println("Maximum value in the array: " + max);
    }
}
```

Output:



Q6. Write a program to find the second largest number in the array.

Input: a[] = {1, 99, 50, 4, 8, 33}
Output: 50

Code:

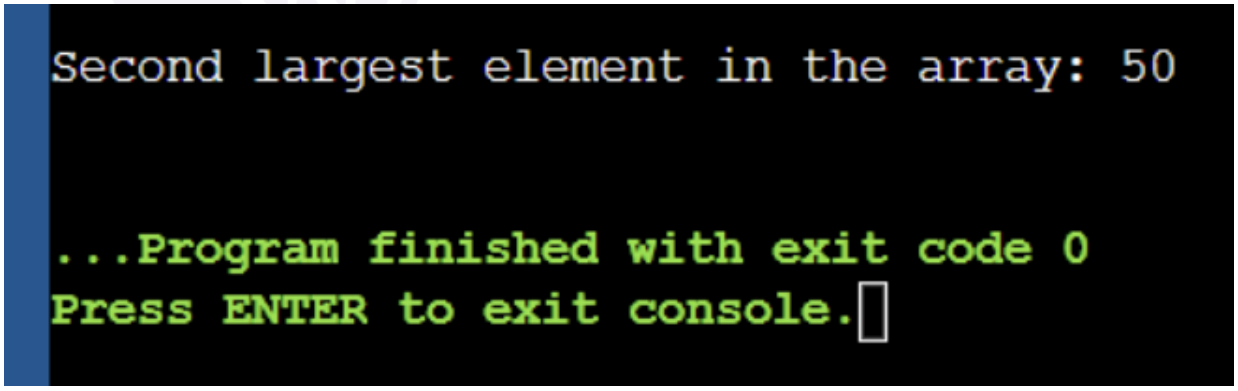
```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 99, 50, 4, 8, 33};

        int firstMax = Integer.MIN_VALUE;
        int secondMax = Integer.MIN_VALUE;

        for (int i = 0; i < arr.length; i++) {
            if (arr[i] > firstMax) {
                secondMax = firstMax;
                firstMax = arr[i];
            } else if (arr[i] > secondMax && arr[i] != firstMax)
{
                secondMax = arr[i];
            }
        }

        if (secondMax != Integer.MIN_VALUE) {
            System.out.println("Second largest element in the
array: " + secondMax);
        } else {
            System.out.println("No second largest element found
in the array.");
        }
    }
}
```

Output:



Q7. MCQ : What is the difference between the 5's in these two expressions?

1. first is particular element, second is type
2. first is array size, second is particular element
3. first is particular element, second is array size
4. both specify array size

Ans. 1. first is particular element, second is array size

Q8. MCQ : What would happen if you assign a value to an element of an array whose subscript exceeds the size of the array?

1. the element will be set to 0
2. nothing, it's done all the time
3. other data may be overwritten
4. error message from the compiler

Ans. 3. other data may be overwritten

Q9. State TRUE or FALSE :

1. The array int num[26] has twenty-six elements.
2. The expression num[1] designates the first element in the array
3. It is necessary to initialize the array at the time of declaration.
4. The expression num[27] designates the twenty-eighth element in the array.

Ans.

1. TRUE
2. FALSE (The expression num[1] designates the second element in the array because array indices start from 0.)
3. FALSE (It's not necessary to initialize the array at the time of declaration, but it's a good practice to do so.)
4. FALSE (The expression num[27] would result in an ArrayIndexOutOfBoundsException because array indices in Java start from 0, so the valid indices for num[26] array would be from 0 to 25.)

Q10. Count the number of elements in given array greater than a given number x.

Input: A[] = {1,2,3,4,5,6,7,8,10} x = 5
Output: 4

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {10, 25, 15, 30, 20};
        int x = 20;

        int count = 0;

        for (int num : arr) {
            if (num > x) {
                count++;
```

```
            }
        }
        System.out.println("Number of elements greater than " + x
+ ": " + count);
    }
}
```

Output: