Experiment Number : 04 **Date: 05.02.2026**

ARRAYS:

Aim:

To understand and implement array operations in Java.

PRE LAB EXERCISE:

QUESTIONS:

✓ **What is an array?**

An array is a data structure that stores multiple values of the same data type in a single variable.

Example:

int marks[5] = {80, 85, 90, 75, 95};

✓ **Why are arrays used?**

- To store multiple values in a single variable
- To reduce the number of variables in a program
- To make data handling easier and organized
- To access elements using index numbers (fast access)
- To perform operations like sorting, searching, and processing data easily
- To improve program efficiency and readability

✓ **What is the difference between array and variable?**

Feature	**Variable**	**Array**
Meaning	Stores only one value	Stores multiple values
Data Storage	Single data item	Multiple data items
Declaration	int x;	int x[5];
Access	Directly by name	By index (x[0], x[1], ...)

Feature	**Variable**	**Array**
Memory	Less memory	More memory
Use	Simple data storage	Bulk data storage

IN LAB EXERCISE:

Objective:

To perform array operations using simple programs.

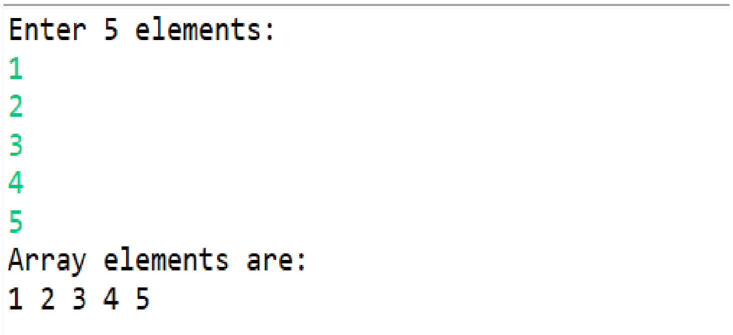
PROGRAMS:

1. Program to Read and Print Array Elements

Code:

```
import java.util.Scanner;
public class ReadPrintArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        System.out.println("Array elements are:");
        for(int i = 0; i < 5; i++)
            System.out.print(arr[i] + " ");
    }
}
```

OUTPUT:



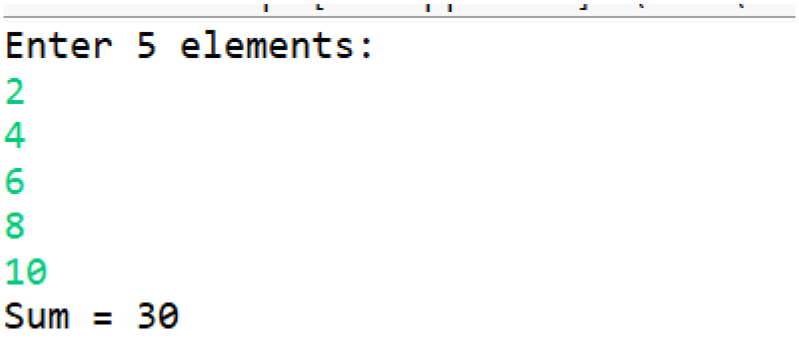
```
Enter 5 elements:
1
2
3
4
5
Array elements are:
1 2 3 4 5
```

2. Program to Find Sum of Array Elements

Code:

```
import java.util.Scanner;
public class SumArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        int sum = 0;
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        for(int i = 0; i < 5; i++)
            sum += arr[i];
        System.out.println("Sum = " + sum);
    }
}
```

OUTPUT:



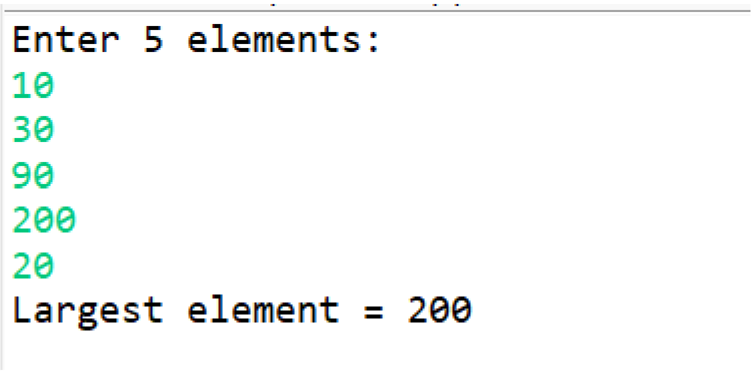
```
Enter 5 elements:
2
4
6
8
10
Sum = 30
```

3. Program to Find Largest Element in an Array

Code:

```
import java.util.Scanner;
public class LargestElement {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        int max = arr[0];
        for(int i = 1; i < 5; i++)
            if(arr[i] > max)
                max = arr[i];
        System.out.println("Largest element = " + max);
    }
}
```

OUTPUT:



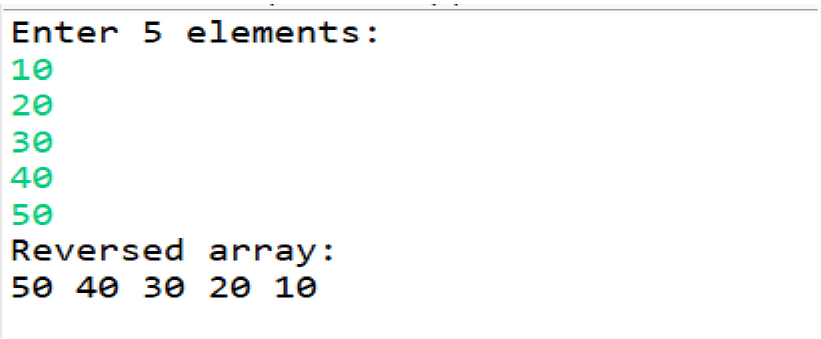
```
Enter 5 elements:
10
30
90
200
20
Largest element = 200
```

4. Program to Reverse an Array

Code:

```
import java.util.Scanner;
public class ReverseArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        System.out.println("Reversed array:");
        for(int i = 4; i >= 0; i--)
            System.out.print(arr[i] + " ");
    }
}
```

OUTPUT:



```
Enter 5 elements:
10
20
30
40
50
Reversed array:
50 40 30 20 10
```

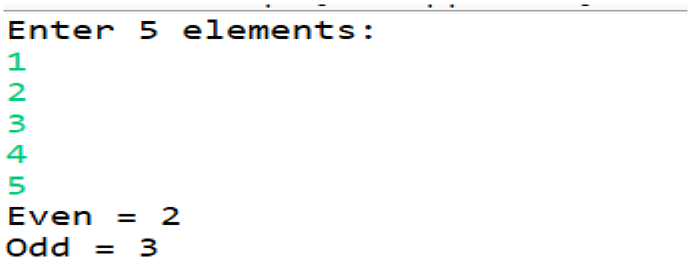
5. Program to Count Even and Odd Numbers

Code:

```
import java.util.Scanner;
public class EvenOddCount {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int[] arr = new int[5];
    int even = 0, odd = 0;
    System.out.println("Enter 5 elements:");
    for(int i = 0; i < 5; i++)
      arr[i] = sc.nextInt();
    for(int i = 0; i < 5; i++) {
      if(arr[i] % 2 == 0)
        even++;
      else
        odd++;
    }

    System.out.println("Even = " + even);
    System.out.println("Odd = " + odd);
  }
}
```

OUTPUT:



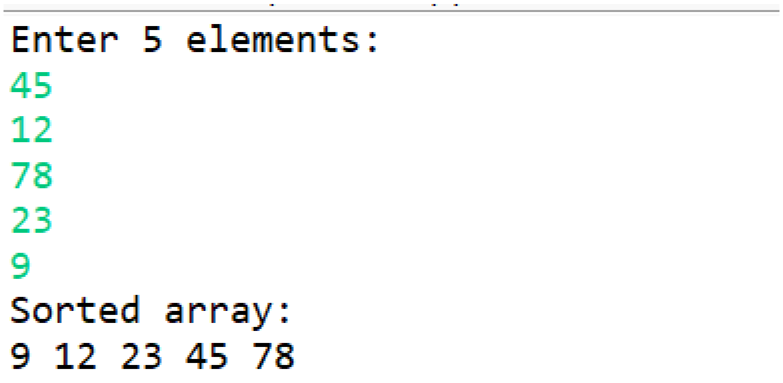
```
Enter 5 elements:
1
2
3
4
5
Even = 2
Odd = 3
```

6. Program to Sort Array in Ascending Order

Code:

```
import java.util.Scanner;
public class SortArray {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        int temp;
        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        for(int i = 0; i < 5; i++) {
            for(int j = i + 1; j < 5; j++) {
                if(arr[i] > arr[j]) {
                    temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
        System.out.println("Sorted array:");
        for(int i = 0; i < 5; i++)
            System.out.print(arr[i] + " ");
    }
}
```

OUTPUT:



```
Enter 5 elements:
45
12
78
23
9
Sorted array:
9 12 23 45 78
```

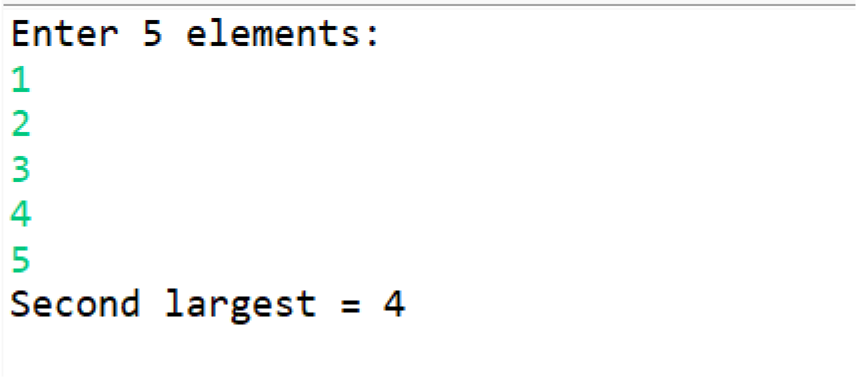
7. Program to Find Second Largest Element

Code:

```
import java.util.Scanner;
public class SecondLargest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];

        System.out.println("Enter 5 elements:");
        for(int i = 0; i < 5; i++)
            arr[i] = sc.nextInt();
        int largest = Interger.MIN_VALUE;
        int second =INTERGER.MIN_VALUE;
        for(int i = 0; i < 5; i++) {
            if(arr[i] > largest) {
                second = largest;
                largest = arr[i];
            }
            else if(arr[i]>second && arr[i]!=largest)
            {
                second=arr[i];
            }
        }
        System.out.println("Second largest = " + second);
    }
}
```

OUTPUT:



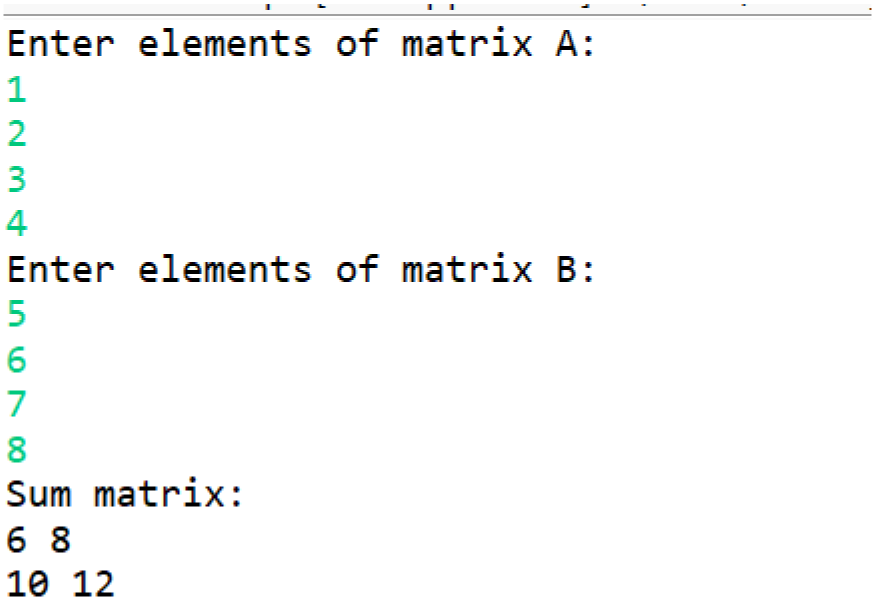
```
Enter 5 elements:
1
2
3
4
5
Second largest = 4
```

8. Program for Matrix Addition (2D Array)

Code:

```
import java.util.Scanner;
public class MatrixAddition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] a = new int[2][2];
        int[][] b = new int[2][2];
        int[][] sum = new int[2][2];
        System.out.println("Enter elements of matrix A:");
        for(int i = 0; i < 2; i++)
            for(int j = 0; j < 2; j++)
                a[i][j] = sc.nextInt();
        System.out.println("Enter elements of matrix B:");
        for(int i = 0; i < 2; i++)
            for(int j = 0; j < 2; j++)
                b[i][j] = sc.nextInt();
        for(int i = 0; i < 2; i++)
            for(int j = 0; j < 2; j++)
                sum[i][j] = a[i][j] + b[i][j];
        System.out.println("Sum matrix:");
        for(int i = 0; i < 2; i++) {
            for(int j = 0; j < 2; j++)
                System.out.print(sum[i][j] + " ");
            System.out.println();
        }
    }
}
```

OUTPUT:



```
Enter elements of matrix A:
1
2
3
4
Enter elements of matrix B:
5
6
7
8
Sum matrix:
6 8
10 12
```

POST LAB EXERCISE:

- ✓ **Why is array indexing usually started from zero instead of one?**

  Array indexing starts from zero because the first element is stored at the base memory address. The index represents the offset from this base address, so using zero makes memory access simpler and faster.

- ✓ **What happens if we try to access an array element outside its declared size?**

  Accessing an array element outside its declared size leads to undefined behavior. It may produce garbage values, cause runtime errors, crash the program, or overwrite other memory locations.

- ✓ **How does memory allocation differ for static arrays and dynamic arrays?**

Feature	**Static Array**	**Dynamic Array**
Memory Allocation	At compile time	At runtime
Size	Fixed	Can be changed
Memory Location	Stack / Static memory	Heap memory
Flexibility	Less flexible	More flexible
Example	int arr[10];	int *arr = malloc(10 * sizeof(int));

- ✓ **Why is searching faster in arrays compared to linked lists?**

  Searching is faster in arrays because they support direct access using index values. In linked lists, elements must be accessed sequentially by traversing nodes, which takes more time.

- ✓ **What is the difference between contiguous and non-contiguous memory allocation?**

Feature	**Contiguous Memory**	**Non-Contiguous Memory**
Memory Layout	Continuous block	Separate scattered blocks
Example	Arrays	Linked Lists
Access Speed	Faster	Slower
Memory Usage	Efficient but fixed	Flexible but less efficient
Addressing	Single base address	Multiple addresses

Result:

Thus the array operations were executed successfully.

ASSESSMENT

Description	**Max Marks**	**Marks Awarded**
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		