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 collection of elements of the same data type stored in contiguous memory locations. Each element in an array is accessed using an index number. Arrays allow storing multiple values under a single variable name. They make data handling more organized and efficient. Arrays can be one-dimensional or multi-dimensional.

- ✓ Why are arrays used?

  Arrays are used to store a large number of related values efficiently. They reduce the need for declaring multiple variables. Arrays make it easier to perform operations like searching, sorting, and traversing data. They also help in writing cleaner and more structured code. Arrays improve memory management and program performance.

- ✓ What is the difference between array and variable?

  A variable stores only a single value at a time, whereas an array can store multiple values. Each variable has a single memory location, but an array uses multiple contiguous memory locations.

  Variables are accessed directly by name, while array elements are accessed using an index. Arrays are suitable for handling bulk data, while variables are used for individual data values.

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] + " ");
    }
}
```

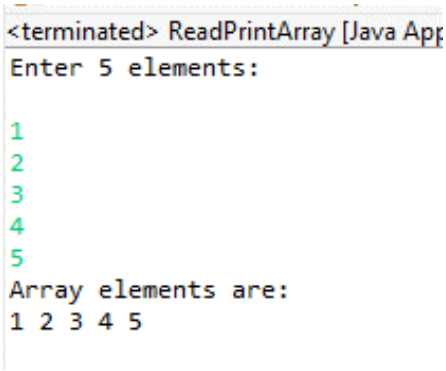
OUTPUT:

Input:
1 2 3 4 5

Output:

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:

Input:
1 2 3 4 5

Output:

Sum = 15

```
<terminated> SumArray [
Enter 5 elements
1
2
3
4
5
Sum 15
```

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:

Input:

3 12 13 13 21

Output:

Largest element = 21

```
<terminated> LargestElement [Java
Enter 5 elements:
3 12 13 13 21
Largest element = 21
```

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] + " ");
    }
}
```

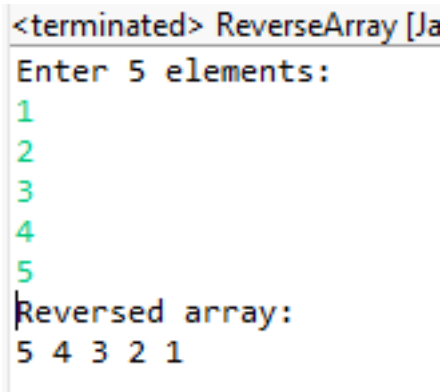
OUTPUT:

Input:
1 2 3 4 5

Output:

Reversed array:

5 4 3 2 1



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:

Input:
1 2 3 4 5

Output:

Even = 2

Odd = 3

```
<terminated> EvenOdd
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:

Input:
4 5 1 2 7

Output:

Sorted array:

1 2 4 5 7

```
<terminated> SortArray [Java Applic
Enter 5 elements:
4
5
1
2
7
Sorted array:
1 2 4 5 7
```

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 = arr[0];
        int second = arr[0];
        for(int i = 0; i < 5; i++) {
            if(arr[i] > largest) {
                second = largest;
```

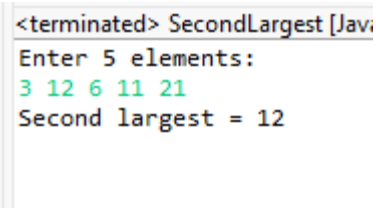
```
                largest = arr[i];
            }
        }
        System.out.println("Second largest = " + second);
    }
}
```

OUTPUT:

Input:
3 12 6 11 13

Output:

Second largest = 12



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:

Matrix A:

1 2

3 4

Matrix B:

5 6

7 8

Sum matrix:

6 8

10 12

```
<terminated> MatrixAddition [Java Appli
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 it represents the offset from the base memory address. The first element is stored at the base address itself, so its offset is 0. This makes address calculation faster and simpler for the compiler. Zero-based indexing also improves performance in low-level memory access. Hence, most programming languages follow this approach.

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

  Accessing an array element outside its declared size causes an error or unexpected behavior. In some languages, it may lead to runtime errors or exceptions. In others like C or C++, it can access invalid memory locations. This may result in incorrect output or program crashes. Therefore, bounds checking is very important.

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

  Static arrays are allocated memory at compile time and have a fixed size. Their size cannot be changed during program execution. Dynamic arrays are allocated memory at runtime and their size can be modified. Dynamic memory allocation allows better memory utilization. However, it requires proper memory management.

- ✓ Why is searching faster in arrays compared to linked lists?
  Arrays allow direct access to elements using index values. This enables faster searching, especially for binary search in sorted arrays. In linked lists, elements must be accessed sequentially. Each node requires traversal from the beginning. Hence, searching in arrays is generally faster than in linked lists.

- ✓ What is the difference between contiguous and non-contiguous memory allocation?
  Contiguous memory allocation stores elements in consecutive memory locations. This allows faster access and better cache performance. Non-contiguous memory allocation stores elements at different memory locations linked together. It provides flexible memory usage but slower access. Linked lists are an example of non-contiguous allocation.

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	