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 and accessed using a single name with an index.

✓ Why are arrays used?

Arrays are used to:

- Store multiple values of the same type in one variable
- Make programs shorter and cleaner
- Allow easy access to data using index numbers
- Help in processing large amounts of data efficiently (like marks, salaries, scores)

✓ What is the difference between array and variable?

A **variable** stores only one value, whereas an **array** stores multiple values.

A **variable** uses a single memory location, whereas an **array** uses multiple contiguous memory locations.

A **variable** is accessed directly by its name, whereas an **array** is accessed using an index.

A **variable** is suitable for single data items, whereas an **array** is suitable for handling grouped or bulk data.

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] + " ");
   }
}
```

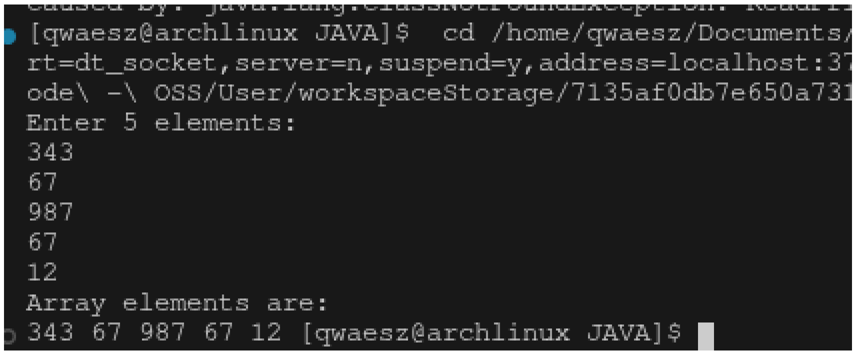
OUTPUT:

Input:

```
343 67 987 67 12
```

Output:

```
Array elements are:
343 67 987 67 12
```


```
[qwaesz@archlinux JAVA]$  cd /home/qwaesz/Documents/
rt=dt_socket,server=n,suspend=y,address=localhost:37
ode\ -\ OSS/User/workspaceStorage/7135af0db7e650a731
Enter 5 elements:
343
67
987
67
12
Array elements are:
343 67 987 67 12 [qwaesz@archlinux JAVA]$
```

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);
    }
}
```

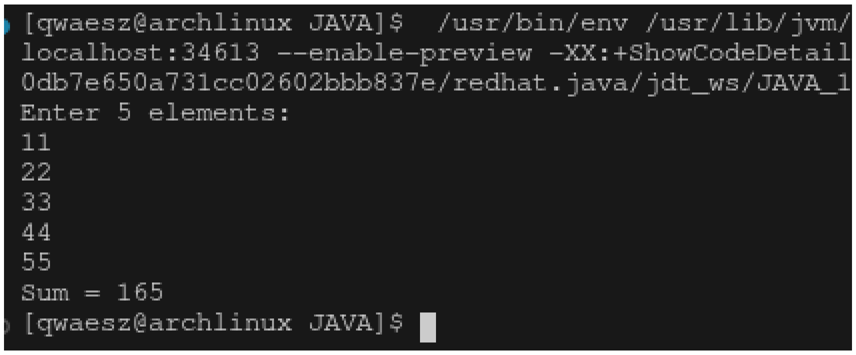
OUTPUT:

Input:

```
11 22 33 44 55
```

Output:

```
Sum = 165
```



```
[qwaesz@archlinux JAVA]$  /usr/bin/env /usr/lib/jvm/
localhost:34613 --enable-preview -XX:+ShowCodeDetail
0db7e650a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_1
Enter 5 elements:
11
22
33
44
55
Sum = 165
[qwaesz@archlinux JAVA]$
```

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);

    }

}
```

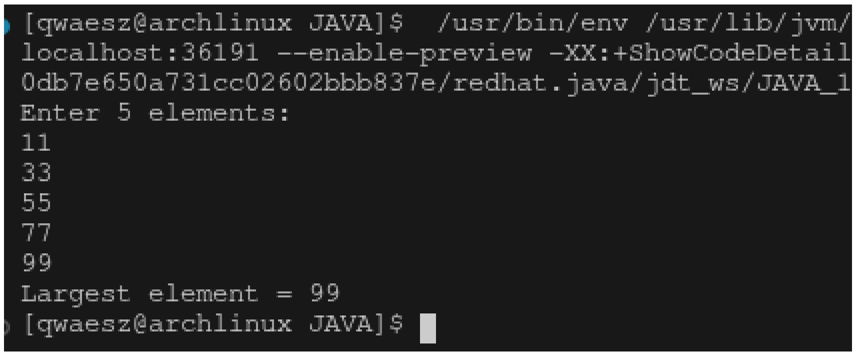
OUTPUT:

Input:

11 33 55 77 99

Output:

Largest element = 99



```
[qwaesz@archlinux JAVA]$  /usr/bin/env /usr/lib/jvm/
localhost:36191 --enable-preview -XX:+ShowCodeDetail
0db7e650a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_1
Enter 5 elements:
11
33
55
77
99
Largest element = 99
[qwaesz@archlinux JAVA]$ 
```

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:

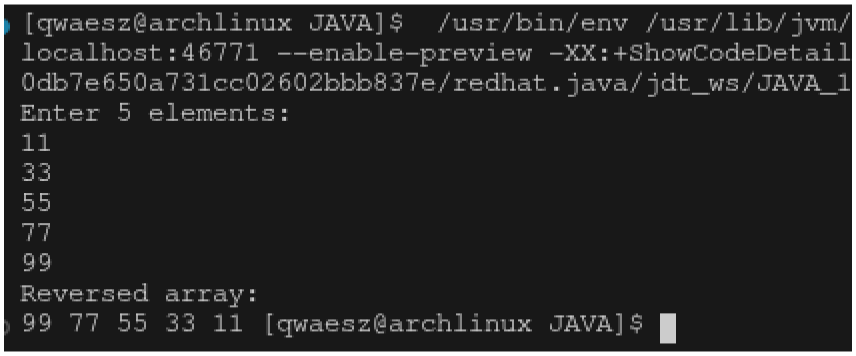
Input:

11 33 55 77 99

Output:

Reversed array:

11 33 55 77 99



```
[qwaesz@archlinux JAVA]$  /usr/bin/env /usr/lib/jvm/
localhost:46771 --enable-preview -XX:+ShowCodeDetail
0db7e650a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_1
Enter 5 elements:
11
33
55
77
99
Reversed array:
99 77 55 33 11 [qwaesz@archlinux JAVA]$ 
```

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:

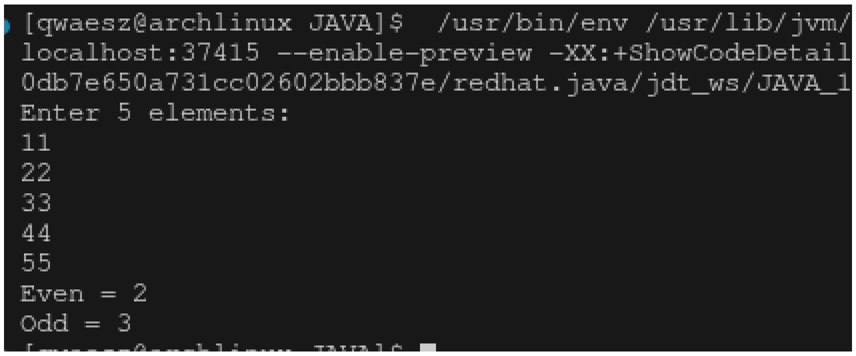
Input:

11 22 33 44 55

Output:

Even = 2

Odd = 3



```
[qwaesz@archlinux JAVA]$  /usr/bin/env /usr/lib/jvm/
localhost:37415 --enable-preview -XX:+ShowCodeDetail
0db7e650a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_1
Enter 5 elements:
11
22
33
44
55
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] + " ");
    }
}
```

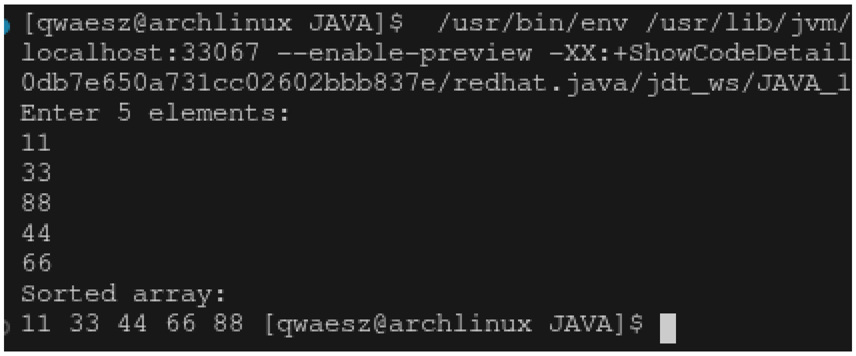
OUTPUT:

Input:
11 33 88 44 66

Output:

Sorted array:

11 33 44 66 88



```
[qwaesz@archlinux JAVA]$  /usr/bin/env /usr/lib/jvm/
localhost:33067 --enable-preview -XX:+ShowCodeDetail
0db7e650a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_1
Enter 5 elements:
11
33
88
44
66
Sorted array:
11 33 44 66 88 [qwaesz@archlinux JAVA]$
```

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);
    }
}
```

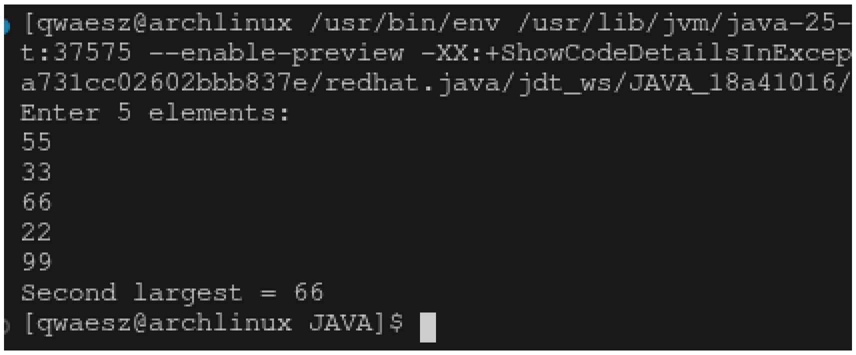
OUTPUT:

Input:

```
55 33 66 22 99
```

Output:

```
Second largest = 66
```



```
[qwaesz@archlinux /usr/bin/env /usr/lib/jvm/java-25-
t:37575 --enable-preview -XX:+ShowCodeDetailsInExcep
a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_18a41016/
Enter 5 elements:
55
33
66
22
99
Second largest = 66
[qwaesz@archlinux JAVA]$ 
```

8. Program for Matrix Addition (2D Array)

Code:

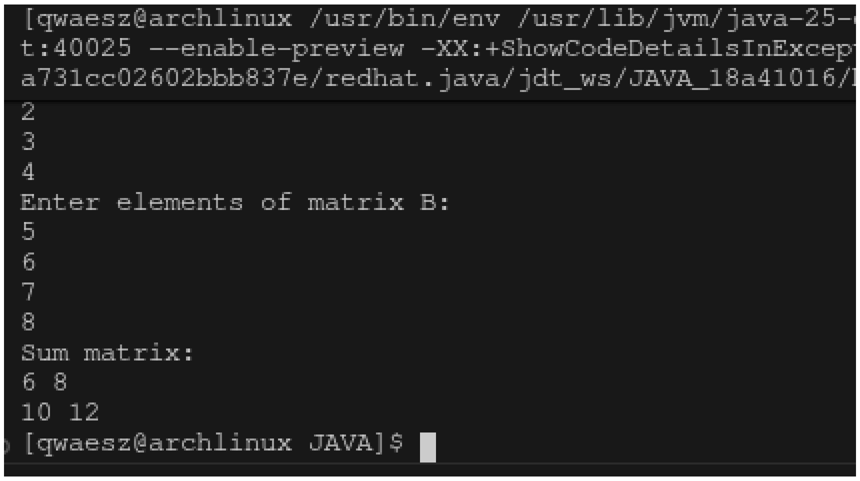
```
import java.util.Scanner;
public class MatrixAddition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] a = new int[2][2];
        int[][] b = new int[2][2];
        int[][] sum = new int[2][2];
        System.out.println("Enter elements of 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
```



```
[qwaesz@archlinux /usr/bin/env /usr/lib/jvm/java-25-
t:40025 --enable-preview -XX:+ShowCodeDetailsInExcep
a731cc02602bbb837e/redhat.java/jdt_ws/JAVA_18a41016/
2
3
4
Enter elements of matrix B:
5
6
7
8
Sum matrix:
6 8
10 12
[qwaesz@archlinux JAVA]$
```

POST LAB EXERCISE

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

  Array indexing starts from zero because the index represents the offset from the starting memory address.

  The first element is stored at the base address, so its offset is 0. This makes address calculation faster and simpler for the system.

- ✓ 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 cause:

  - Garbage value access
  - Program crash or runtime error
  - Memory corruption

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

  - Static arrays have a fixed size decided at compile time.
  - Static arrays are stored in stack or static memory.
  - Dynamic arrays get memory allocated at runtime.
  - Dynamic arrays use heap memory.
  - Static arrays cannot be resized, while dynamic arrays can be resized.

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

  - Arrays store elements in contiguous memory locations, allowing direct access using index.
  - Linked lists require sequential traversal from the head node, making searching slower.

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

  - Contiguous memory allocation stores elements in adjacent memory locations.
  - It allows faster access to elements.
  - Non-contiguous memory allocation stores elements at different memory locations.

- Accessing elements takes more time due to pointer traversal.
- Arrays use contiguous memory, while linked lists use non-contiguous memory.

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		