1.What do u mean by an array

Sol- In theory, an array is a data structure that stores a collection of elements, typically of the same data type, in a contiguous block of memory. The elements are stored in a way that allows for efficient access using an index or key.

Here’s a bit more detail:

Fixed size: The size of an array is usually defined when it’s created. It can’t easily change in some programming languages (although in others, like dynamic arrays or lists, they can grow and shrink).

Indexing: Each element in the array is identified by its position or index. In most languages, arrays are zero-indexed, meaning the first element is at index 0, the second at index 1, and so on.

Homogeneous data: In most cases, all elements in an array are of the same data type (e.g., all integers, all strings, etc.). However, in some languages, you can have arrays with mixed types.

Efficient access: Since arrays use contiguous memory locations, accessing an element by its index is fast (usually constant time, O(1)).

Example in theory:

If we have an array that stores the ages of 5 people:

Copy

```
[25, 30, 22, 28, 35]
```

Here, each number represents an age, and you can access each one using its index. The first element (25) is at index 0, the second (30) is at index 1, and so on.

In short, an array is a way to organize multiple items into a single, accessible structure.

2. How to create an array

Sol-

In Java, you can create an array in several ways. Here's how to do it:

1. Declaring and initializing an array with values:

You can declare an array and initialize it with values at the same time.

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

2. Declaring an array first and then initializing it:

You can declare the array without values and then assign values to the elements individually.

Int[] numbers = new int[5]; // declares an array of size 5 numbers[0] = 1; numbers[1] = 2; numbers[2] = 3; numbers[3] = 4; numbers[4] = 5;

3. Declaring and initializing an array of objects:

For arrays of objects, you follow a similar pattern but use the object type instead of a primitive type.

String[] names = {“Alice”, “Bob”, “Charlie”};

String[] names = new String[3]; // declare array

names[0] = “Alice”; names[1] = “Bob”; names[2] = “Charlie”;

4. Multi-dimensional arrays:

You can create arrays with multiple dimensions (like matrices or tables).

2D array:

Int[][] matrix = new int[3][3]; // creates a 3x3 matrix matrix[0][0] = 1; matrix[0][1] = 2; matrix[0][2] = 3;

Or, initializing directly:

Int[][] matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

3.Can we change the size of array at runtime

Sol- In Java, arrays are of a fixed size once they are created, meaning you cannot directly change the size of an array at runtime.

4.Can u declare an array without assigning the size of array

In Java, we cannot directly declare an array without specifying its size at the time of creation. Java arrays have a fixed size once they’re initialized. However, you can use other structures like Array List if you need a dynamic array-like object that doesn’t require a size declaration.

5.What is the default value of array

Sol-

In Java, the default value of an array depends on the type of elements the array holds:

For arrays of primitive types:

Int[] → 0

Double[] → 0.0

Char[] → ‘\u0000’ (the null character)

Boolean[] → false

Byte[] → 0

Float[] → 0.0f

Long[] → 0L

Short[] → 0

For arrays of reference types (e.g., String[], Object[]):

The default value is null for each element.

So, if you declare an array without initializing its elements explicitly, the elements will be set to these default values based on their type.

Example:

```
Int[] intArray = new int[5];  // Default values: [0, 0, 0, 0, 0]
```

```
Boolean[] boolArray = new boolean[3];  // Default values: [false, false, false]

String[] strArray = new String[4];  // Default values: [null, null, null, null]
```

The default values are automatically set when the array is created.

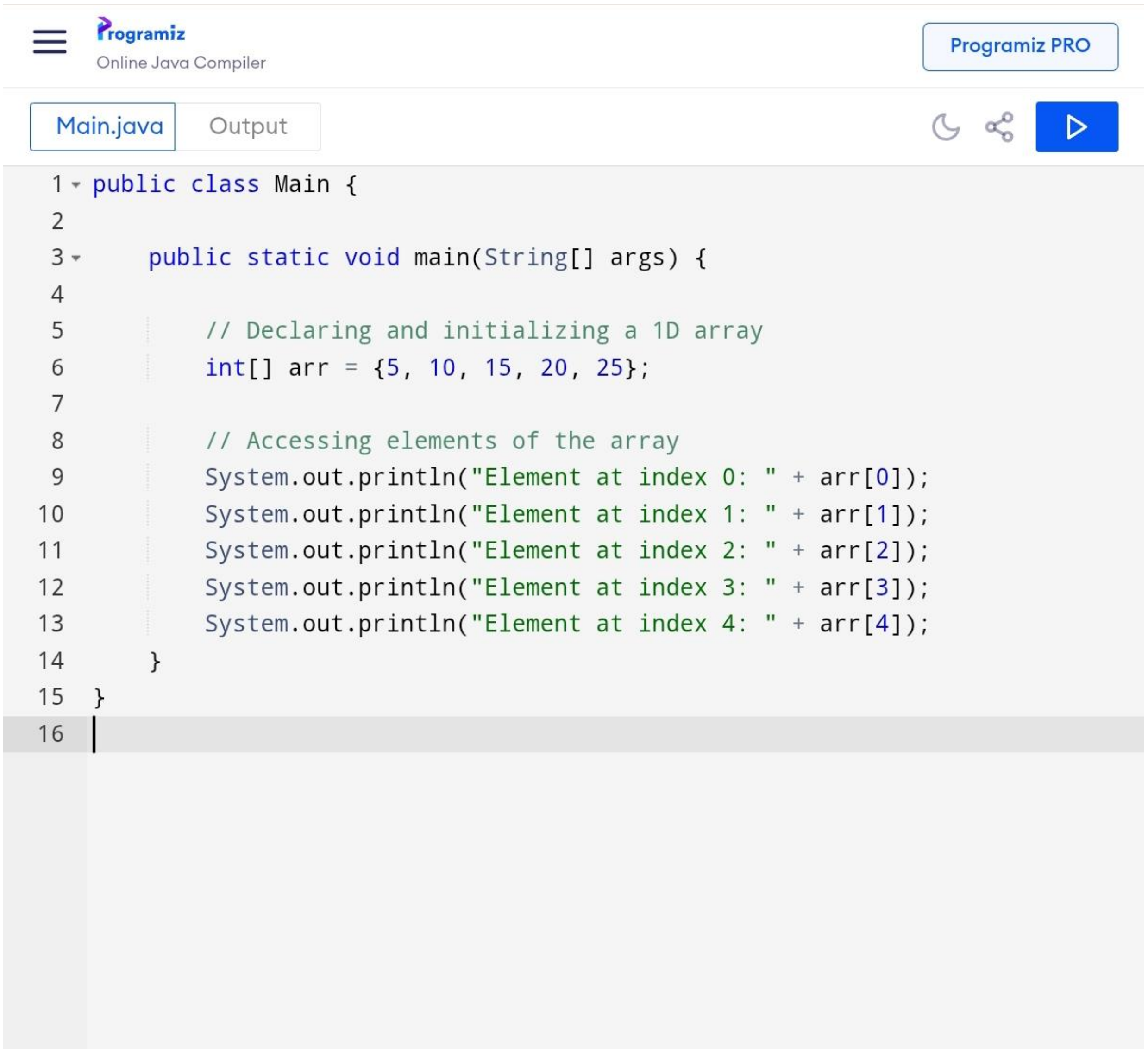
6.**What is a 1D array with an example**

Sol- In Java, a 1D array is a data structure that stores a fixed-size sequence of elements of the same type. You can think of it as a list of values, where each value is accessed by an index.

Example of a 1D Array in Java:

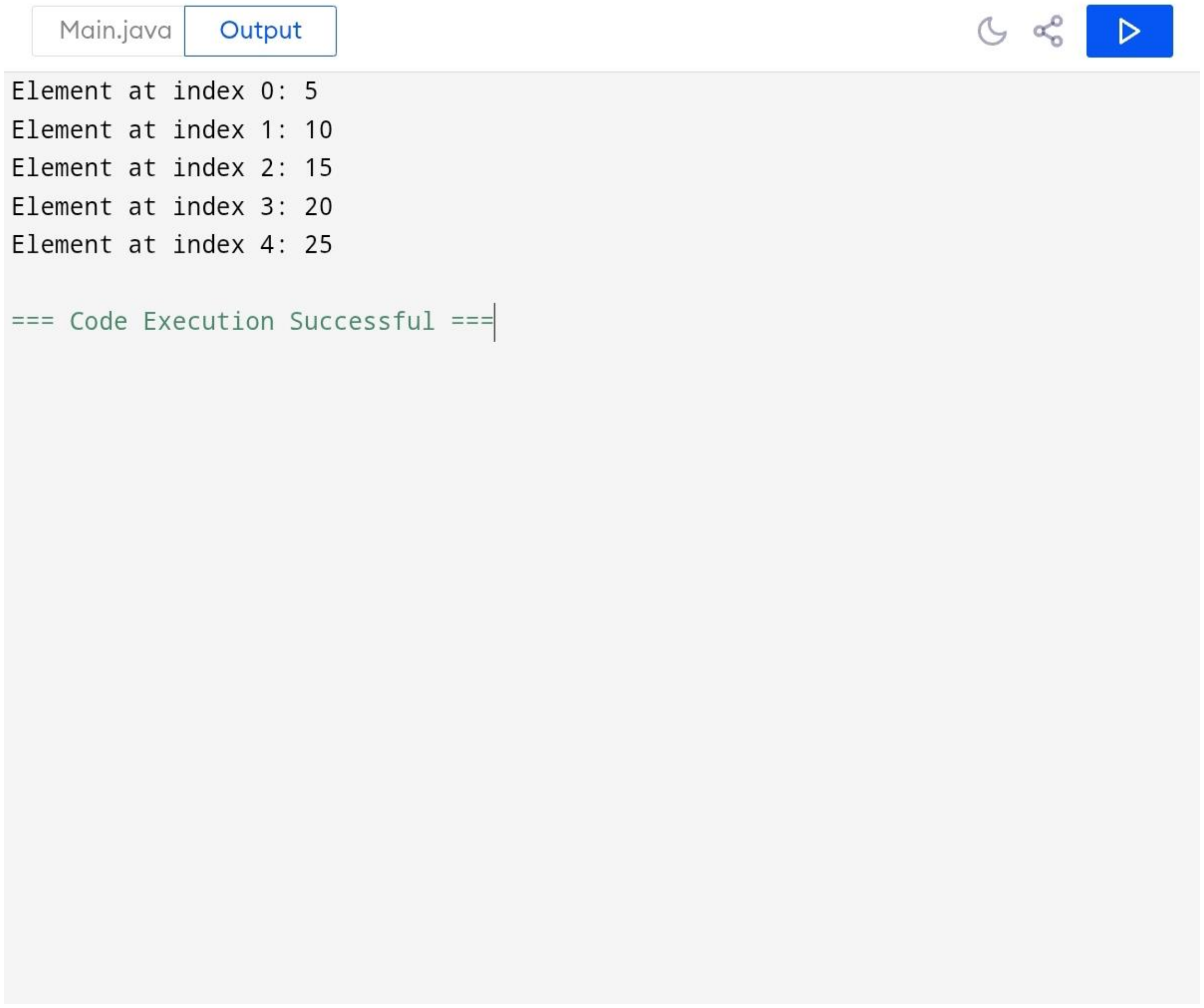
```
Public class Main
{
  Public static void main(String[] args)
{
    // Declaring and initializing a 1D array
    Int[] arr = {5, 10, 15, 20, 25};

    // Accessing elements of the array
    System.out.println("Element at index 0: " + arr[0]);
    System.out.println("Element at index 1: " + arr[1]);
    System.out.println("Element at index 2: " + arr[2]);
    System.out.println("Element at index 3: " + arr[3]);
    System.out.println("Element at index 4: " + arr[4]);
  }
}
```

Screenshot of above code



```java
public class Main {

    public static void main(String[] args) {

        // Declaring and initializing a 1D array
        int[] arr = {5, 10, 15, 20, 25};

        // Accessing elements of the array
        System.out.println("Element at index 0: " + arr[0]);
        System.out.println("Element at index 1: " + arr[1]);
        System.out.println("Element at index 2: " + arr[2]);
        System.out.println("Element at index 3: " + arr[3]);
        System.out.println("Element at index 4: " + arr[4]);
    }
}
```

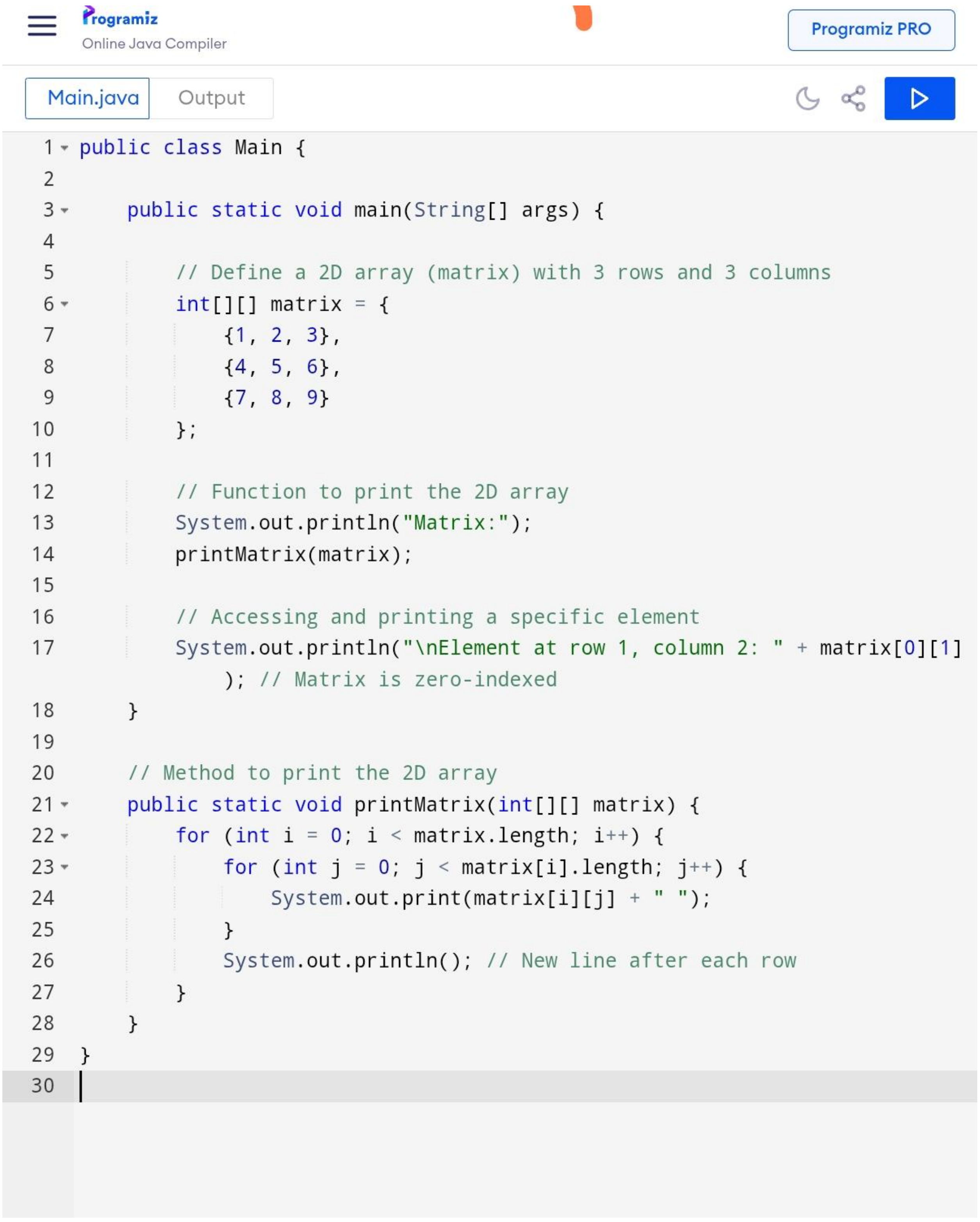
Output screenshot



```
Element at index 0: 5
Element at index 1: 10
Element at index 2: 15
Element at index 3: 20
Element at index 4: 25

=== Code Execution Successful ===
```

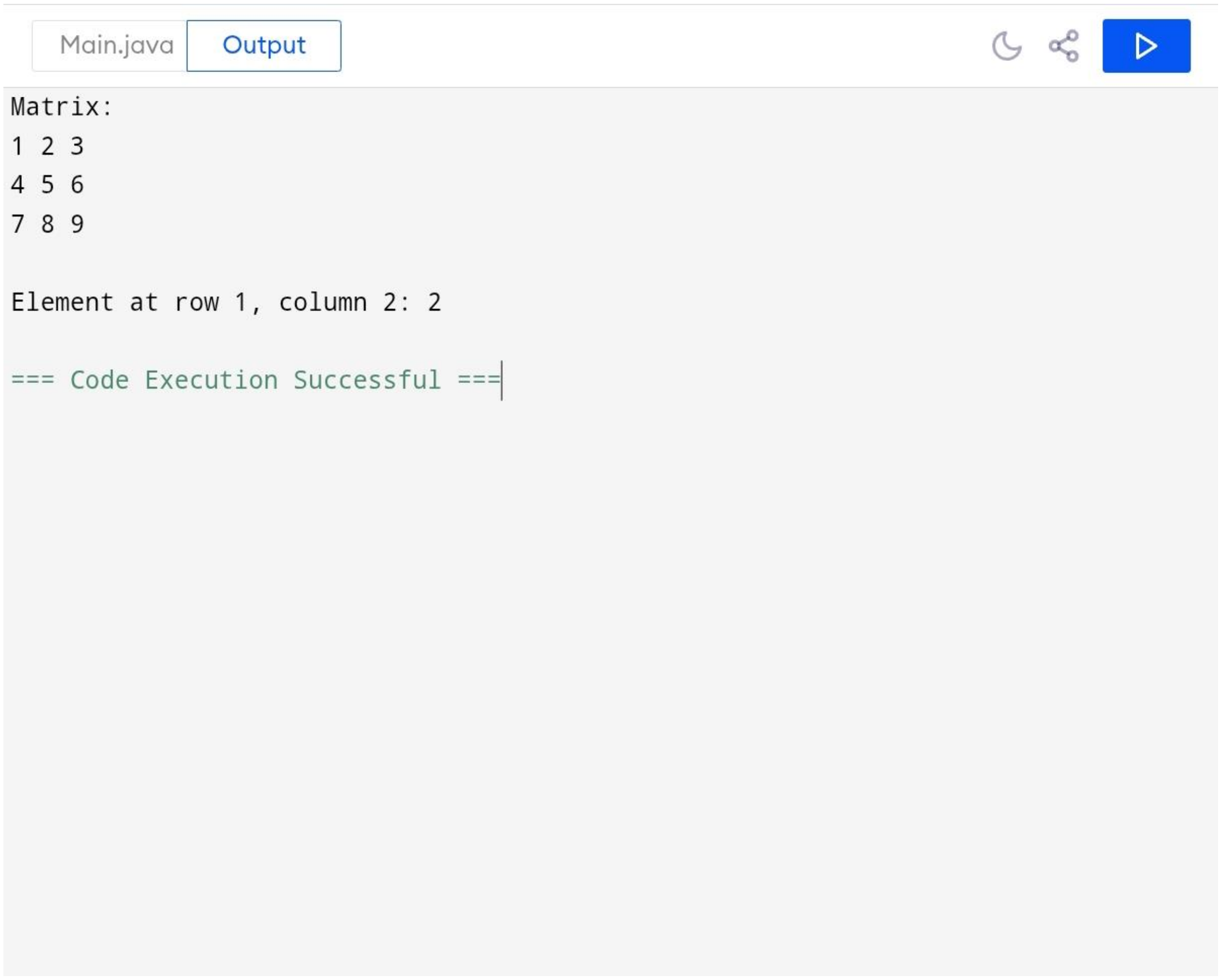
7.Write a program on 2d array

Sol-



```java
public class Main {

    public static void main(String[] args) {

        // Define a 2D array (matrix) with 3 rows and 3 columns
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        // Function to print the 2D array
        System.out.println("Matrix:");
        printMatrix(matrix);

        // Accessing and printing a specific element
        System.out.println("\nElement at row 1, column 2: " + matrix[0][1]
            ); // Matrix is zero-indexed
    }

    // Method to print the 2D array
    public static void printMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println(); // New line after each row
        }
    }
}
```

Output



```
Matrix:
1 2 3
4 5 6
7 8 9

Element at row 1, column 2: 2

=== Code Execution Successful ===
```