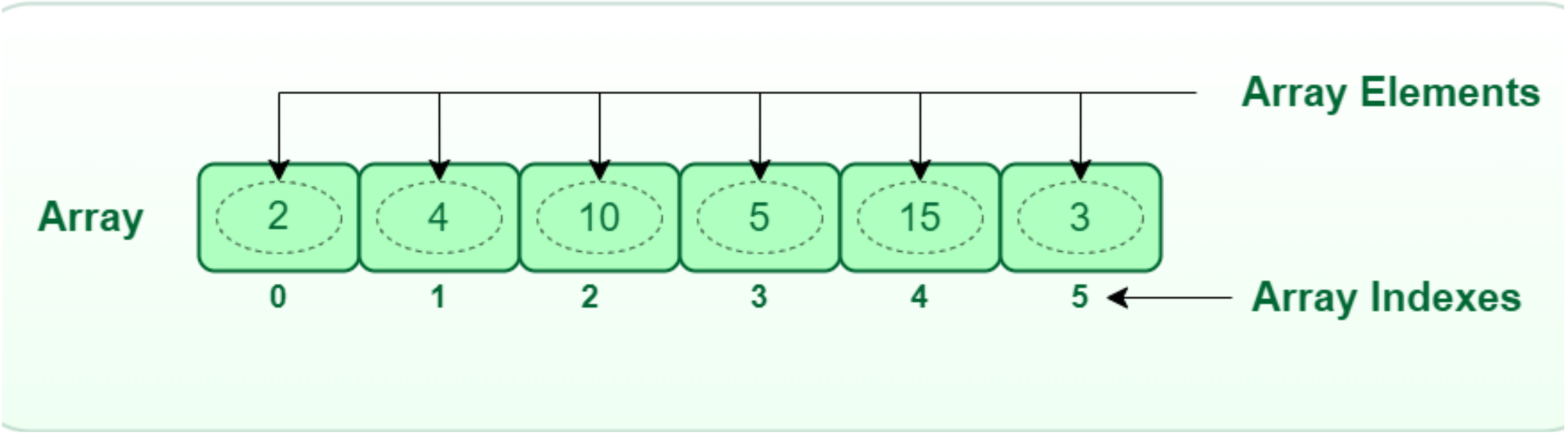
Array and Strings Unit 1

.

Definition

- An **array** in Java is a data structure used to store multiple values of the same type in a single variable.
- Arrays are fixed in size, and each element can be accessed using an index.
- Elements are stored in contiguous memory locations.
- Indexing starts at 0.



Syntax to declare an array:

1. Using `new` keyword with size:

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

2. Using `new` keyword with initialization:

```
int[] arr = new int[] {1, 2, 3};
```

3. Direct initialization:

```
int[] arr = {1, 2, 3};
```

4. Multidimensional arrays:

```
int[][] matrix = new int[3][3];
```

[ ] arr vs arr[ ]

: Both `int[] arr` and `int arr[]` are valid.

int []arr = new arr[3]

int arr[] = new arr[3] ..... both are valid

Example:

Part 1: int a[],b;

a = new int[4];

b = 10;

Part 2: int []a,b;

a = new int[4];

b = 10;

Types of Arrays in Java

1. Single-Dimensional Arrays:

   - A simple list of elements stored in a linear format.

2. Multi-Dimensional Arrays:

   - Used to store data in rows and columns (like a matrix).

   Declaration : int[ ][ ] matrix = new int[3][3]; // 3x3 matrix

3. Jagged Arrays:
   - A special type of multi-dimensional array where rows can have different sizes.

Cont..

.

- Declaration Jagged Array:

```
int[][] jaggedArray = new int[3][ ];
jaggedArray[0] = new int[2];
jaggedArray[1] = new int[3];
jaggedArray[2] = new int[1];
```

- Default values: **int: 0,**

  char: '\u0000', boolean: false, objects: null

.

java.util.Arrays Class

- Java provides utility classes in the java.util package for array manipulation.
- **Commonly Used Methods:**

1. Arrays.toString(array): Converts an array to a string.

Example: int[] numbers = {1, 2, 3};

System.out.println(Arrays.toString(numbers)); // Output: [1, 2, 3]

2**. Arrays.fill(array, value):** Fills the array with the specified value.

Example: int[] arr = new int[5];

Arrays.fill(arr, 7);

System.out.println(Arrays.toString(arr)); // Output: [7, 7, 7, 7, 7]

Commonly Used Methods Array Class

3**. binarySearch(array, key):** Searches for the specified key in a sorted array and returns the index.

- Example: int[] arr = {1, 2, 3, 4, 5};

  System.out.println(Arrays.binarySearch(arr, 3)); **// Output: 2**

4. equals(array1, array2): Checks if two arrays are equal.

Example : int[] arr1 = {1, 2, 3};

int[] arr2 = {1, 2, 3};

System.out.println(Arrays.equals(arr1, arr2)); // Output: true

5. copyOf(array, newLength): Creates a copy of the array with the specified new length.

Example: int[] arr = {1, 2, 3};

int[] newArr = Arrays.copyOf(arr, 5);

System.out.println(Arrays.toString(newArr)); // Output: [1, 2, 3, 0, 0]

Cont..

.

6. **asList(array):** Converts the array into a List.

Example: String[] arr = {"A", "B", "C"};

List<String> list = Arrays.asList(arr);

System.out.println(list); // Output: [A, B, C]

7. **Arrays.sort(array):** Sorts the array in ascending order.