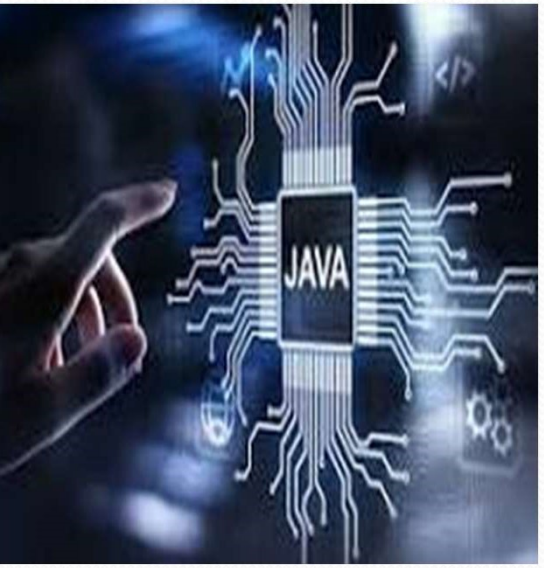
SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution
Coimbatore-35

Department of Computer Science & Engineerin

23ITT202 – Object Oriented Programming

I B.E CSE/ II SEMESTER



UNIT II :Control Statements and Constructo

Topic : Arrays, methods and attributes in Java

Topics for Discussion

- ✓ Introduction to Arrays.
- ✓ What is an Array?
- ✓ Why Arrays are Used.
- ✓ Types of Arrays
- ✓ Declaration & Initialization
- ✓ Examples
- ✓ Assignment

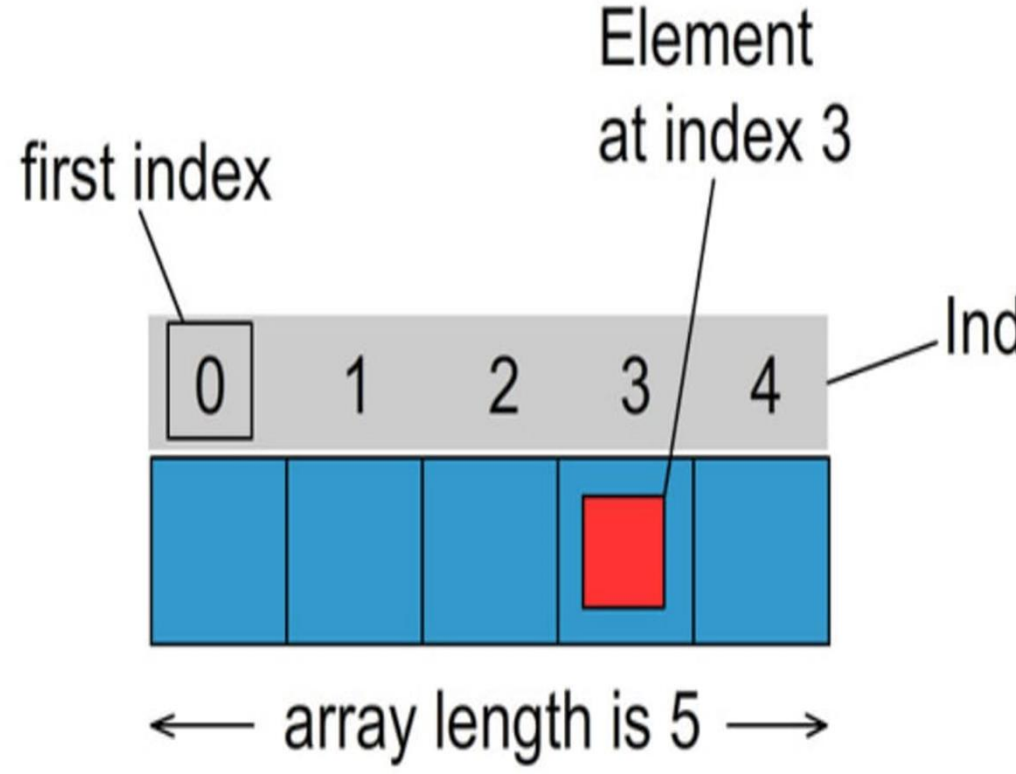
INTRODUCTION TO ARRAYS

Array is a collection of similar data types.

Stores multiple values using a single variable name.

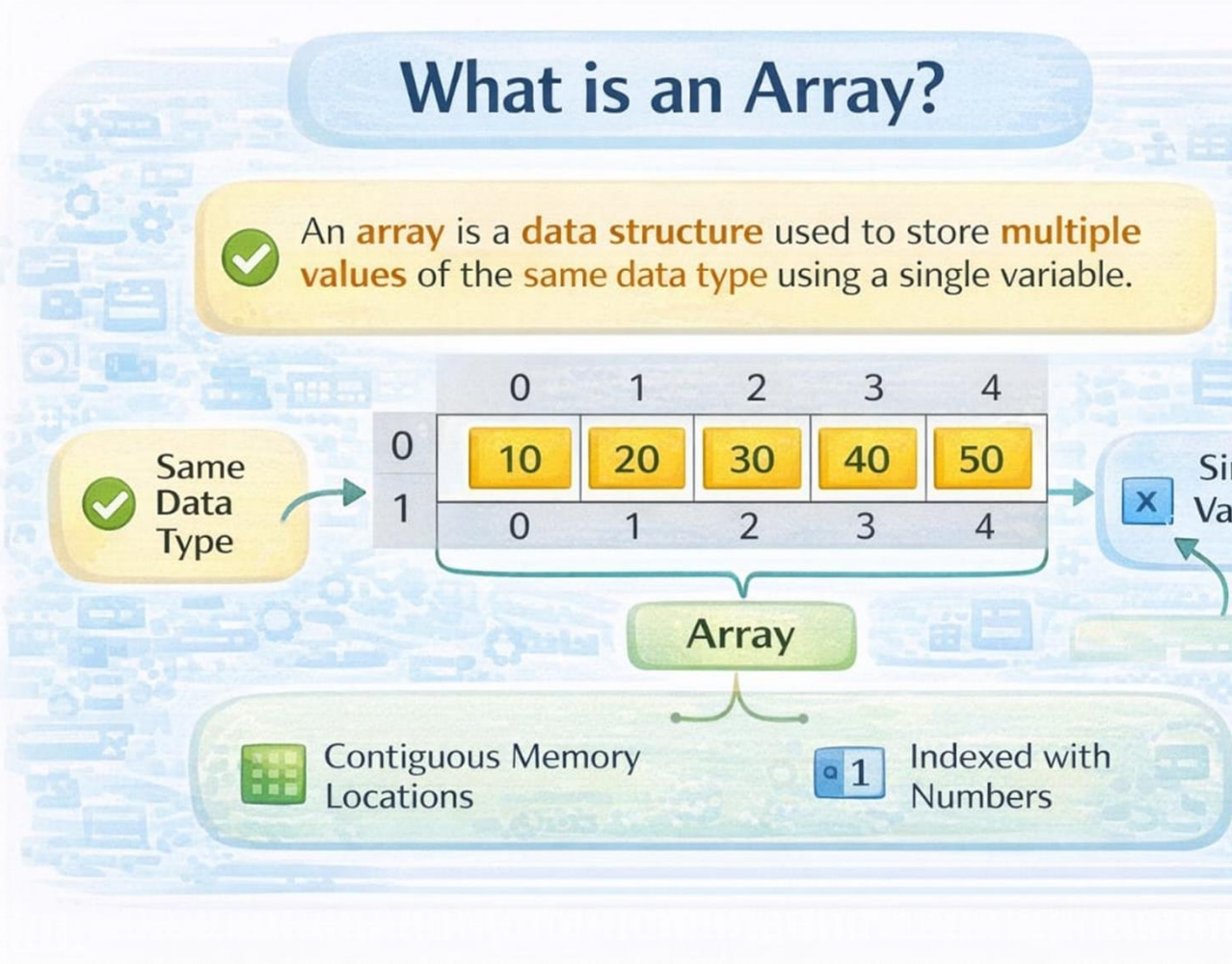
Each value is stored at a specific index.

Index starts from 0 in Java



What Is An Array?

- An array is a data structure used to store multiple values of the same data type.
- Values are stored in contiguous memory locations.
- Each element is accessed using an index number



Why Array is Important?

Arrays are **important** because they:

- Store **multiple values** using a **single** variable.
- Reduce the need for **many separate variables**.
- Make data handling easy and organized.
- Improve program efficiency and readability.

Array

- Contiguous Memory Locations
- Indexed with Numbers

Types Of Arrays In Java

One-Dimensional Array

Stores data in a single line

Uses one index

xample: int[] a = {1, 2, 3};

2. Two-Dimensional Array

- Stores data in rows and colur
- Uses two indices

Example: int[][] a = {{1,2},{3,4

✔ **1D** → simple list
✔ **2D** → table / matrix

Types of Arrays in Java

Java supports mainly two types of arrays:

ne-dimensional Array

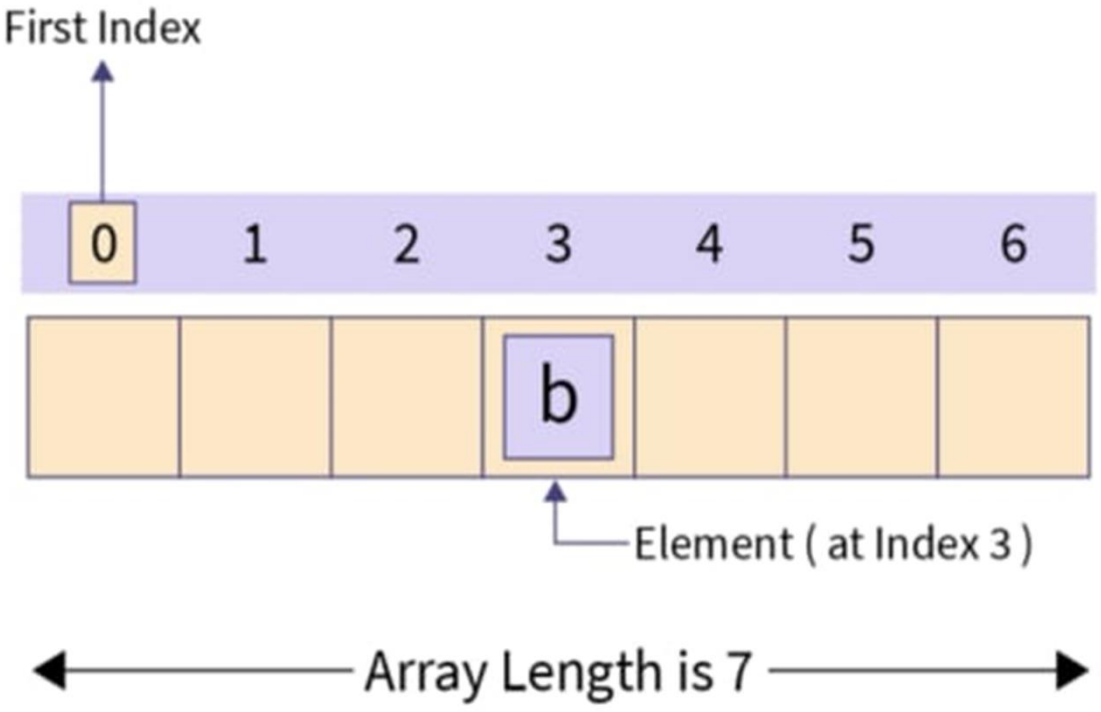
Stores elements in a single row.

Accessed using one index.

Simple and commonly used

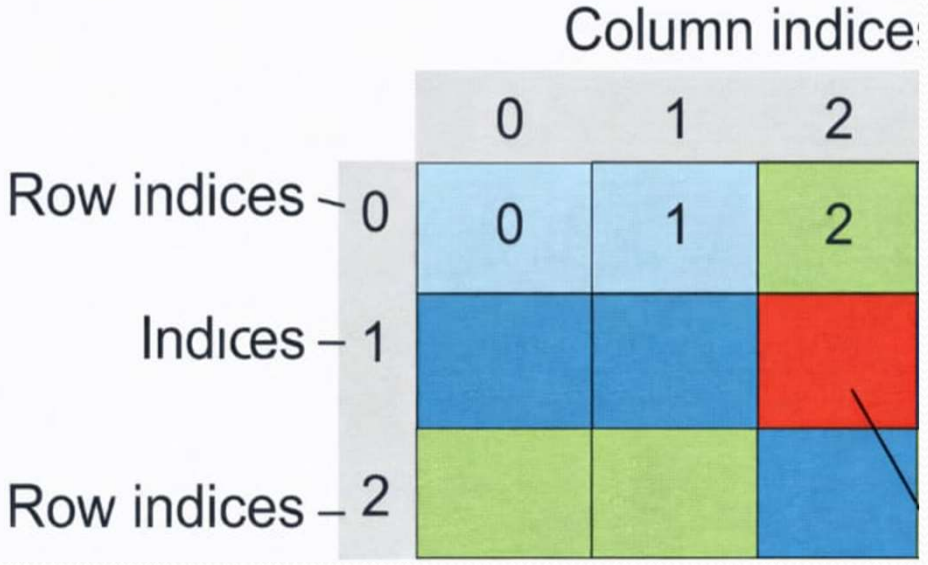
ntax:

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



vo-dimensional Array

- Stores data in rows and columns.
- Accessed using two indexes.
- Used to represent tables or matrices.



ntax:

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

2D ARRAY

```
import java.io.*;

class Main {
    public static void main(String[] args){

        // Array Intialised and Assigned
      int[][] arr = { { 1, 2 }, { 3, 4 } };

        // Printing the Array
      for (int i = 0; i < 2; i++){
         for (int j = 0; j < 2; j++)
           System.out.print(arr[i][j]+" ");
         System.out.println();
         }    }}
```

```
Output:

1 2
3 4
```

eclaration & Initialization

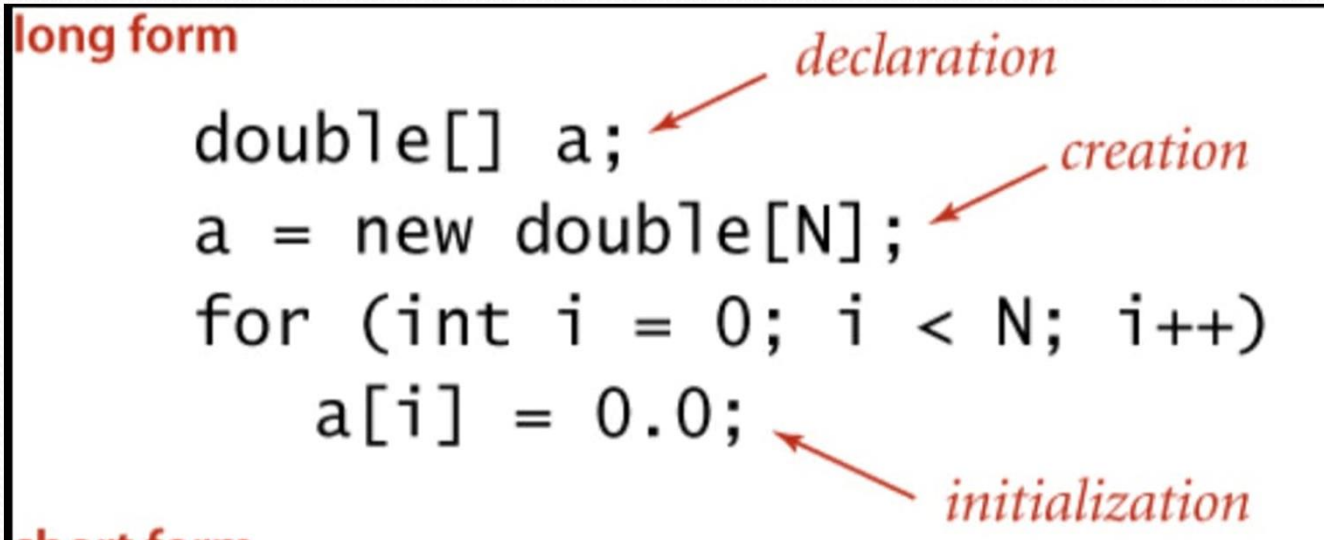
Declaration: Defines array type.

Initialization: Assigns values.

yntax:

int a[] = {10, 20, 30, 40};

long form



short form

```
double[] a = new double[N];
```

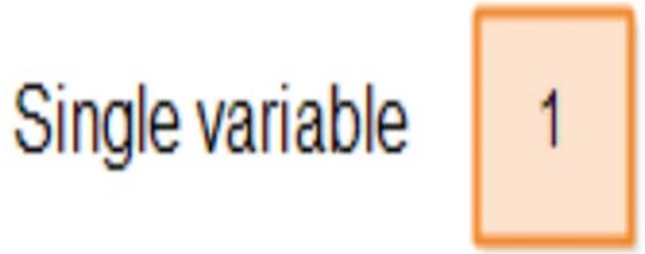
initializing declaration

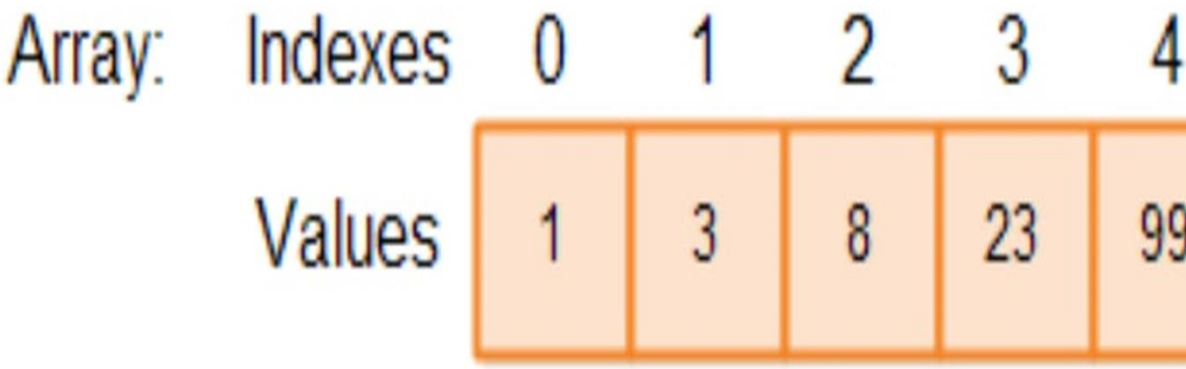
```
int[] a = { 1, 1, 2, 3, 5, 8
```

ccessing Array Elements

Array elements accessed using index.

ndex starts from 0





xample:

```
System.out.println(a[0]);
```

Sample program1

- **Explanation:**
- **Array** → int[] marks = {85, 72, 90, 66, 58};
  - Stores multiple values in a single variable.
  - Accessed using index numbers (0 to length-1).
- **Switch Case**
  - Used to perform different operations based on user choice.
  - Each case performs a different array operation.
  - break prevents execution from continuing to other cases.

```
 java.util.Scanner;
class ArraySwitchExample {
static void main(String[] args) {
== ARRAY DECLARATION =====
y to store 5 student marks
arks = {85, 72, 90, 66, 58};
er sc = new Scanner(System.in);
lay menu
.out.println("===== MENU =====");
.out.println("1. Display All Marks");
.out.println("2. Display Highest Mark");
.out.println("3. Display Lowest Mark");
.out.println("4. Display Average Marks");
.out.println("Enter your choice (1-4): ");
ice = sc.nextInt();
== SWITCH CASE =====
 (choice) {
```

```
case 1:
// Display all elements of array
System.out.println("Student Marks:");
for (int i = 0; i < marks.length; i++) {
System.out.println("Mark " + (i + 1) + ": " + marks[i]);
}
break;
case 2:
// Find highest mark
int max = marks[0];
for (int i = 1; i < marks.length; i++) {
if (marks[i] > max) {
max = marks[i];
}}
System.out.println("Highest Mark: " + max);
break;
case 3:
// Find lowest mark
int min = marks[0];
for (int i = 1; i < marks.length; i++) {
if (marks[i] < min) {
min = marks[i];}}
System.out.println("Lowest Mark: " + min);
break;
```

```
case 4:
// Calculate average
int sum = 0;
for (int i = 0; i < marks.length; i++) {
sum += marks[i];
}
double average = (double) sum / marks.length;
System.out.println("Average Marks: " + average);
break;

default:
System.out.println("Invalid Choice!");
}

sc.close();
}
}
```

Sample Output (Example)

```
===== MENU =====
1. Display All Marks
2. Display Highest Mark
3. Display Lowest Mark
4. Display Average Marks
Enter your choice (1-4): 2
Highest Mark: 90
```

Control structures with array – perform bubble sort for given elements

- Problem Statement: You are a logistics manager at an e-commerce warehouse. After receiving a shipment of products, the items need to be sorted based on their weights in ascending order so that lighter weights are packed first and the heaviest items are packed last. Use an basic algorithm to implement the above and do the sorting

The algorithm's logic follows these steps:

- **Iterate through the array:** Use an outer loop that runs from the first element to the second-to-last element. This loop controls the number of passes.
- **Compare adjacent elements:** Inside the outer loop, an inner loop compares each pair of adjacent elements.
- **Swap if out of order:** If the current element is greater than the next element (for ascending order), swap them using a temporary variable.
- **Reduce the comparison range:** After each full pass, the largest unsorted element is guaranteed to be in its correct position at the end of the unsorted portion. The inner loop's range is reduced in subsequent passes to avoid comparing elements that are already sorted.
- **Stop when sorted (Optimization):** An optimization can be added using a boolean flag to check if any swaps occurred in a pass. If no swaps happen during an entire pass, the array is already sorted, and the algorithm can terminate early

Sample program2

ram:

```
ic class Bubble {
 void bubbleSort(int[] arr)

 = arr.length;
nt i = 0; i < n - 1; i++) {
nt j = 0; j < n - i - 1; j++) {
r[j] > arr[j + 1]) {
ap arr[j] and arr[j + 1]
mp = arr[j];
 = arr[j + 1];
+ 1] = temp;
```

```
public static void main(String[] args) {
int[] arr = {64, 34, 25, 12, 22, 11, 90};
System.out.println("Array Before Bubble Sort:");
for (int i =0;i<arr.length;i++)
{
System.out.print(i + " ");
}
System.out.println();
bubbleSort(arr);
System.out.println("Array After Bubble Sort:");
for (int i =0;i<arr.length;i++)
{
System.out.print(i + " ");
} } }
```

Methods and attributes

- **Explanation:**
- **Attributes** → name, rollNumber, and marks
  - These variables store the data of the student.
- **Methods** → setDetails(), displayDetails(), and calculateGrade()
  - Methods define the behavior of the class.
  - They perform actions using the attributes.
- This program clearly shows:
- Attributes hold data.
- Methods perform operations on that data.
- Objects (like s1) use both attributes and methods.

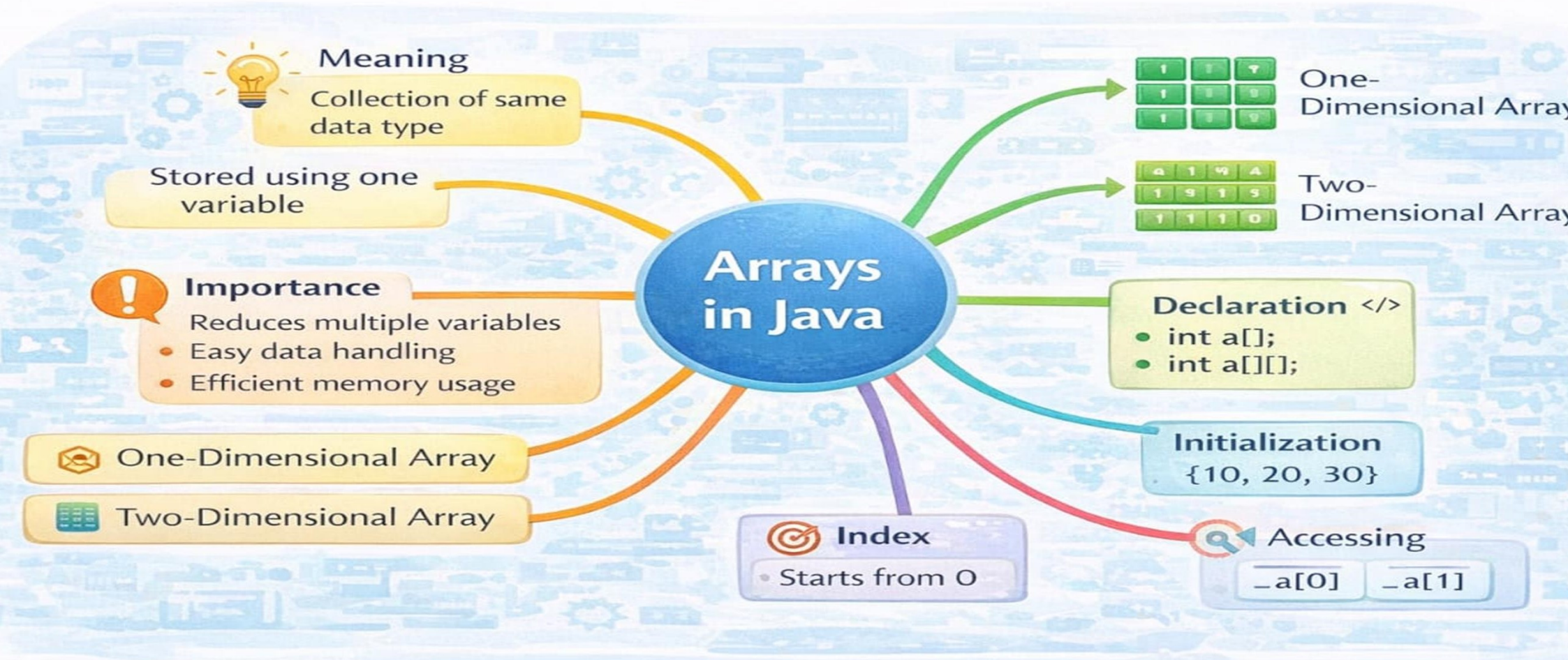
```
ogram to demonstrate Attributes and Methods in Java
efine a class
 Student {

'TRIBUTES (Data Members / Variables)
tributes store the data of the object

g name;                    // Student name
llNumber;                  // Student roll number
le marks;                  // Student marks

=== METHOD 1 =====
ethod to assign values to attributes
setDetails(String n, int r, double m) {
e = n;                     // Assign name
lumber = r;                // Assign roll number
s = m;                     // Assign marks

=== METHOD 2 =====
ethod to display student details
displayDetails() {
m.out.println("Student Name: " + name);
m.out.println("Roll Number: " + rollNumber);
m.out.println("Marks: " + marks);
```

```
// ===== METHOD 3 =====
// Method to calculate grade based on marks
void calculateGrade() {
if (marks >= 90) {
System.out.println("Grade: A");
} else if (marks >= 75) {
System.out.println("Grade: B");
} else if (marks >= 50) {
System.out.println("Grade: C");
} else {
System.out.println("Grade: Fail");
}}}
// Main class
public class MethodsAndAttributesDemo {
public static void main(String[] args) {
// Create an object of Student class
Student s1 = new Student();
// Call method to set values
s1.setDetails("Arun", 101, 85.5);

// Call method to display details
s1.displayDetails();

// Call method to calculate grade
s1.calculateGrade();
}}
```

Output:

```
Student Name: Arun
Roll Number: 101
Marks: 85.5
Grade: B
```

Mind Map


Arrays in Java
Meaning
Collection of same data type
Stored using one variable
Importance
Reduces multiple variables
Easy data handling
Efficient memory usage
One-Dimensional Array
Two-Dimensional Array
Index
Starts from 0
Declaration
int a[];
int a[][];
Initialization
{10, 20, 30}
Accessing
a[0]
a[1]
One-Dimensional Array
Two-Dimensional Array

Assessment

. **The index of an array in Java starts from __.**

nswer: 0

. **Choose the correct answer.**

. **Which of the following is the correct way to declare an array?**

) int a;

) **int a[];**

) array int a;

) int = a[];

. **Write the syntax to declare and initialize an array.**

nswer : int a[] = {10, 20, 30};

References

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
- https://www.javatpoint.com/java-array
- https://www.geeksforgeeks.org/arrays-in-java/

Thank You 😊