| **✅ Important Topics to Learn in Arrays**  **Topic** |  |
| --- | --- |
| 1. Traversal | Using loops like for, for-each, while |
| 2. Input/Output | Taking array input from user and printing it |
| 3. Sum / Average | Calculating sum and average of elements |
| 4. Searching | Linear Search, Binary Search |
| 5. Sorting | Bubble Sort, Selection Sort, Insertion Sort |
| 6. Reverse | Reverse an array |
| 7. 2D Arrays | Matrix format arrays |
| 8. Array vs ArrayList | Difference and when to use what |

### ****Java Arrays – Topic 2: Traversing Arrays with Loops****

(Student Edition Notes)

### ****Definition – Traversing an Array****

### Traversing an array means **visiting each element in the array one by one** to **read**, **print**, or **modify** it.

💡 **Why Important?**  
Almost every real program works with multiple values.  
Loops make it **easy** to process all elements without writing repetitive code.

### ****2. Why Use Loops with Arrays?****

Example without a loop:

int[] marks = {90, 80, 70, 85};

System.out.println(marks[0]);

System.out.println(marks[1]);

System.out.println(marks[2]);

System.out.println(marks[3]);

❌ This is **tedious**, **error-prone**, and **not scalable**.

✅ With a loop:

for (int i = 0; i < marks.length; i++) {

System.out.println(marks[i]);

}

Only **3 lines** of loop code can print **any size array**.

### ****3. The**** for ****Loop (Classic Index-Based Loop)****

A loop that uses a **counter variable (index)** to visit each element.

📜 **Syntax:**

for (int i = 0; i < array.length; i++) {

// Access array element

System.out.println(array[i]);

}

### ****Example 1: Print all elements****

int[] numbers = {10, 20, 30, 40, 50};

for (int i = 0; i < numbers.length; i++) {

System.out.println("Index " + i + ": " + numbers[i]);

}

| **i** | **numbers[i]** |
| --- | --- |
| 0 | 10 |
| 1 | 20 |
| 2 | 30 |
| 3 | 40 |
| 4 | 50 |

### ****Advantages of for loop:****

* Knows **index number**
* Can **modify specific elements**
* Can traverse **in reverse** or **skip elements**

### ****4. The**** for-each ****Loop (Enhanced For Loop)****

### A simplified loop that **directly reads the value** of each element without using an index.

📜 **Syntax:**

for (dataType variable : array) {

// variable = value of current element

}

### ****Example 2: Print all elements using for-each****

int[] numbers = {10, 20, 30, 40, 50};

for (int value : numbers) {

System.out.println(value);

}

💡 **Note:**

* Shorter & cleaner
* Good for **reading** values
* Cannot access element index directly

### ****5. Mini Program – Print Even Numbers****

public class EvenNumbers {

public static void main(String[] args) {

int[] arr = {13, 22, 17, 40, 19, 26};

System.out.println("Even numbers in the array:");

for (int num : arr) {

if (num % 2 == 0) {

System.out.println(num);

}

}

}

}

📌 **Logic:**

* num % 2 == 0 → checks if the number is divisible by 2
* Only prints numbers that are even

### ****6. Common Mistakes While Traversing Arrays****

❌ Forgetting .length and hardcoding size → leads to **ArrayIndexOutOfBoundsException**  
❌ Using wrong loop condition (e.g., i <= arr.length) → also out of bounds  
❌ Modifying array in for-each loop without care

### ****7. Practice Problems****

1. Print all elements of {5, 10, 15, 20, 25} using a for loop
2. Print all elements of {2, 4, 6, 8, 10} using a for-each loop
3. Find and print all odd numbers in {11, 14, 17, 20, 23, 26}
4. Calculate and print the sum of {1, 2, 3, 4, 5}
5. Print elements of {100, 200, 300} in reverse order using a for loop

### ****8. Key Points to Remember****

* .length → property to get array size
* **for loop** → when you need index control
* **for-each loop** → when you just need values
* Always ensure loop limits are **within array size**