Arrays in Java

Definition:

- An **array** is an object that stores multiple elements of the same data type.
- Elements are accessed by indices starting at **0** up to **size-1**.

Types of Arrays in Java

Type	Description
Single Dimensional	Represents elements in one row/level.
Multi Dimensional	Represents elements in multiple rows/levels (2D, 3D, etc.).

Single Dimensional Arrays

1. Declare and Initialize

- **Syntax:**

  ```
  java
  CopyEdit
  DataType[] refVar = {val1, val2, ..., val_n};
  ```

- **Example:**

  ```
  java
  CopyEdit
  int[] ints = {10, 20, 30, 40, 50};
  ```

2. Declare then Initialize

- **Syntax:**

  ```
  java
  CopyEdit
  DataType[] refVar = new DataType[size];
  refVar[0] = val1;
  refVar[1] = val2;
  // ...
  refVar[size-1] = val_n;
  ```

- **Example:**

  ```
  java
  CopyEdit
  int[] ints = new int[5];
  ints[0] = 10;
  ```

```
ints[1] = 20;
ints[2] = 30;
ints[3] = 40;
ints[4] = 50;
```

Additional Details

- **Array Size:**
  Use the `length` variable to get the size.

  ```
  java
  CopyEdit
  System.out.println(ints.length); // prints 5
  ```

- **Iterating Over an Array:**
  Using a standard for loop:

  ```
  java
  CopyEdit
  for (int index = 0; index < ints.length; index++) {
      System.out.println(ints[index]);
  }
  ```

  Using a for-each loop:

  ```
  java
  CopyEdit
  for (int element : ints) {
      System.out.println(element);
  }
  ```

- **Sorting Example (Ascending Order):**
  To sort the elements of an array, you can use the `Arrays.sort()` method:

  ```
  java
  CopyEdit
  import java.util.Arrays;

  public class SortArray {
      public static void main(String[] args) {
          int[] ints = {50, 20, 40, 10, 30};
          Arrays.sort(ints);
          System.out.println("Sorted Array: ");
          for (int num : ints) {
              System.out.print(num + " ");
          }
      }
  }
  ```

  This code sorts the array in ascending order and prints the result.

- **Error Handling:**
  Accessing an element with an index outside `0` to `length-1` results in a `java.lang.ArrayIndexOutOfBoundsException`.

Multi Dimensional Arrays

1. Declare and Initialize

- **Syntax:**

  ```
  java
  CopyEdit
  dataType[][] refVar = { {val1, val2, ...}, {val1, val2, ...}, ... };
  ```

- **Example:**

  ```
  java
  CopyEdit
  int[][] ints = { {1, 2, 3}, {2, 3, 4}, {3, 4, 5} };
  ```

2. Declare then Initialize

- **Syntax:**

  ```
  java
  CopyEdit
  dataType[][] refVar = new dataType[rows][columns];
  refVar[0][0] = val1;
  // ...
  refVar[rows-1][columns-1] = valN;
  ```

- **Example:**

  ```
  java
  CopyEdit
  int[][] ints = new int[3][3];
  ints[0][0] = 1;
  ints[0][1] = 2;
  ints[0][2] = 3;

  ints[1][0] = 2;
  ints[1][1] = 3;
  ints[1][2] = 4;

  ints[2][0] = 3;
  ints[2][1] = 4;
  ints[2][2] = 5;
  ```

Additional Details

- **Array Size:**
  - `ints.length` gives the number of rows.
  - `ints[0].length` gives the number of columns in the first row.
  - Example:

    ```
    java
    ```

```
CopyEdit
System.out.println(ints.length);      // 3
System.out.println(ints[1].length);     // 3
System.out.println(ints[0].length + ints.length); // 3 + 3 = 6
System.out.println(ints[2][2]);         // 5
```

- **Iterating Over a 2D Array:**
  Using nested for loops:

  ```
  java
  CopyEdit
  for (int i = 0; i < ints.length; i++) {
      for (int j = 0; j < ints[i].length; j++) {
          System.out.print(ints[i][j] + " ");
      }
      System.out.println();
  }
  ```

  Using a for-each loop:

  ```
  java
  CopyEdit
  for (int[] row : ints) {
      for (int val : row) {
          System.out.print(val + " ");
      }
      System.out.println();
  }
  ```

- **Error Handling:**
  Accessing an index outside the array's valid range will throw a `java.lang.ArrayIndexOutOfBoundsException`.

Anonymous Arrays

- **Definition:**
  Anonymous arrays are used to pass arrays as parameters directly without needing a separate declaration.
- **Syntax for Declaration:**

  ```
  java
  CopyEdit
  int[] refVar = new int[]{val1, val2, ..., val_n};
  ```

- **Example with Method Parameter:**

  ```
  java
  CopyEdit
  public class Test {
      public static float aggregateMarks(int[] smarks) {
          int addResult = 0;
          for (int val : smarks) {
              addResult = addResult + val;
  ```

```
        }
        float aggregate = addResult / (float) smarks.length;
        return aggregate;
    }
    public static void main(String[] args) {
        // Option 1: Declare an anonymous array and pass its
reference
        int[] smarks = new int[]{67, 89, 91, 78, 82, 76};
        float aggregate = aggregateMarks(smarks);
        System.out.println("Aggregate Marks    : " + aggregate);

        // Option 2: Pass the anonymous array directly
        float aggregateDirect = aggregateMarks(new int[]{67, 89, 91,
78, 82, 76});
        System.out.println("Aggregate Marks    : " +
aggregateDirect);
    }
}
```

- **Note:**
  Anonymous arrays are best used when dealing with a small number of elements.