

Arrays

Java Reference

1-D arrays

An array holds a fixed number of values of one type. Index from 0, and get the size with `.length`.

```
public class Main {  
    public static void main(String[] args) {  
        int[] scores = {88, 71, 95};  
        System.out.println(scores[0]);           // 88  
        System.out.println(scores.length);       // 3  
        scores[1] = 100;  
        System.out.println(scores[1]);           // 100  
    }  
}
```

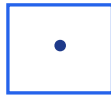
`int n = 7;`



n

a primitive holds
its value directly

`int[] a = {88, 71, 95};`



a (a reference)



[0]

[1]

[2]

a.length
= 3

the array object

06-array-memory

A primitive holds its value; an array variable holds a reference to the array object

Array algorithms

Walk the array with a loop to find a max, a total, a count, or to search . A for-each loop (`for (int x : a)`) reads each value in turn.

```
public class Main {
    public static void main(String[] args) {
        int[] a = {3, 9, 2, 7};
        int max = a[0], total = 0;
        for (int x : a) {
            if (x > max) max = x;
            total += x;
        }
        System.out.println(max + " " + total);    // 9 21
    }
}
```

2-D arrays

A 2-D array is a grid of rows and columns: `grid[row][col]`.

```
public class Main {
    public static void main(String[] args) {
        int[][] grid = {{1, 2, 3}, {4, 5, 6}};
        System.out.println(grid[1][2]);    // 6
        for (int[] row : grid) {
            for (int v : row) System.out.print(v + " ");
        }
        System.out.println();              // 1 2 3 4 5 6
    }
}
```

Common mistakes

- An array's size is `a.length` (no brackets, no `()`), and it is fixed when created.
- Valid indexes are 0 to `a.length - 1`; `a[a.length]` throws `ArrayIndexOutOfBoundsException`.
- A new `int[5]` is filled with zeros, not left empty.