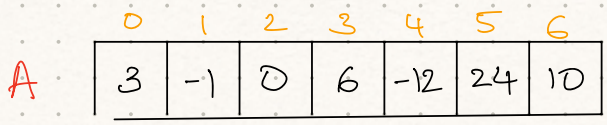
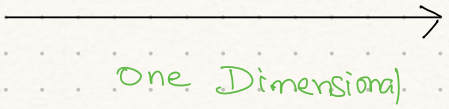
Multi-Dimensional Arrays

- What are multi-dimensional arrays
- Why needed ?
- Java Syntax
- Problems.

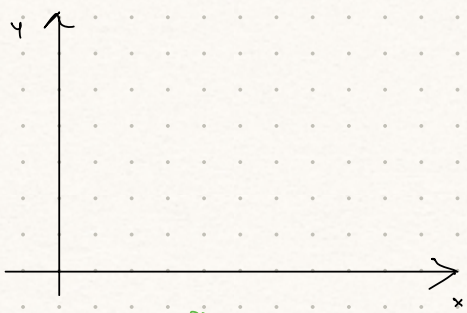


Array of integers

1 - Dimensional (1D) arrays.

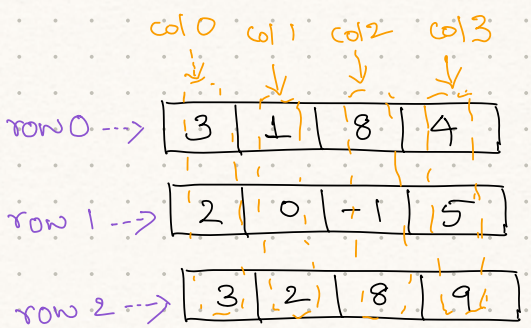


One Dimensional



Two dimensional

(add one more dimension to 1D)



3 rows , 4 columns.

$$\begin{bmatrix} 3 & 1 & 8 & 4 \\ 2 & 0 & -1 & 5 \\ 3 & 2 & 8 & 9 \end{bmatrix}$$

3 x 4 Matrix

↑ # rows ↑ # columns.

In general, a 2-D array:

m × n array

m : number of rows

n : number of columns.

m & n can be different

rows ↓ columns ↓

2 × 2

3 × 2

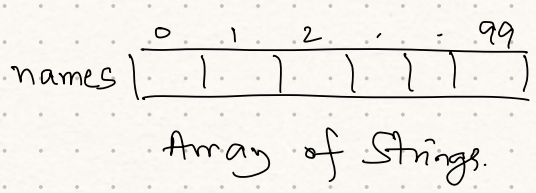
4 × 4

Why needed ?

Stock Price

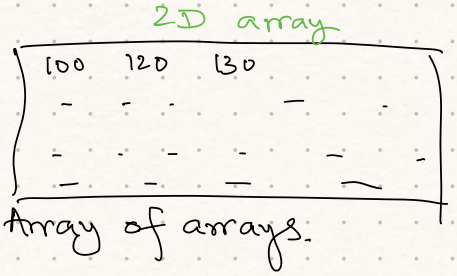
	MON	TUE	WED	THU	FRI	SAT
GOOG	100	120	130	80	95	100
AMAZ	120	180	250	300	350	400
FACE	1000	950	600	800	750	800
APPLE	150	150	170	180	160	200

```
String name0 = "";
String name1 = "";
String name2 = "";
        .
        .
String name99 = "";
```



Array of Strings.

```
int[] arr0;
int[] arr1;
        .
        .
int[] arr999;
```



Array of arrays.

```
String[][] names = {
                    {"Anu", "Bob", "Cathy"},
                    {"Daria", "Erza", "Fani"},
                    {"Gowri", "Haneef", "Ira"}
                   };
```

Anu	Bob	Cathy
Daria	Erza	Fani
Gowri	Haneef	Ira.

Dimension : 3 x 3 array of Strings.

Array Indexing :

names	0	1	2 ← cols
0	Anu (0 0)	Bob (0 1)	Cathy (0 2)
1	Daria (1 0)	~~Erza~~ Harish (1 1)	Fani (1 2)
2	Gowri (2 0)	Haneef (2 1)	Ira. (2 2)

↓ rows

names[1][1] = "Harish";

names[2][3]

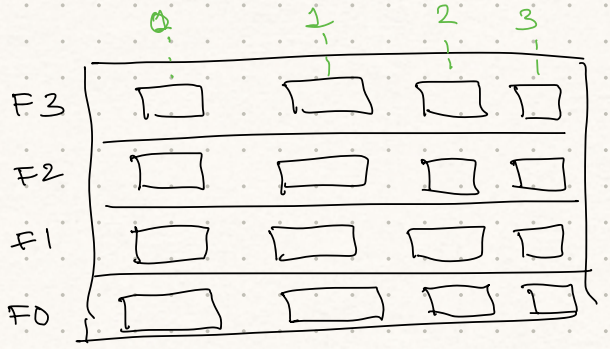
row ↓ col ↓

names[1][0] // Daria

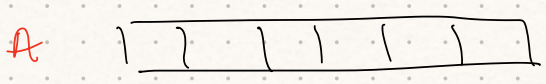
names[2][1] // Haneef

names[0][0] // Anu

In general A[i][j] refers to the element in row i & column j.



Finding Dimension



elements: A.length

2D array 'arr'

arr.length = # rows.

arr[0].length = # columns

```
int[][]
arr = { {1, 2, 3, 4},   <- arr[0]
        {5, 6, 7, 8},   <- arr[1]
        {9, 10, 11, 12} <- arr[2]
      };
```

A	0	1	2	3	4
0	3 (0,0)	1 (0,1)	5 (0,2)	2 (0,3)	1 (0,4)
1	4 (1,0)	0 (1,1)	-4 (1,2)	1 (1,3)	2 (1,4)
2	5 (2,0)	4 (2,1)	-5 (2,2)	3 (2,3)	1 (2,4)
3	8 (3,0)	2 (3,1)	-1 (3,2)	0 (3,3)	1 (3,4)

m = 4
n = 5

Output:

3 1 5 2 1
4 0 -4 1 2
5 4 -5 3 1
8 2 -1 0 1

Dimensions: m × n array.

Print A[i][j] (row, col)

```
m = A.length;  // rows
n = A[0].length;  // cols.
for (int i=0; i<m; i++) {
    for (int j=0; j<n; j++) {
        S.o.Print(A[i][j] + " ");
    }
    S.o.println();
}
```

```
for ( int i=0; i<m; i++ ) {
    for(int j=0; j<n; j++) {
        S.o.Print(A[i][j]+" ");
    }
    S.o.println();
}
```

Outputs: 3 1 5 2 1
4 0 -4

A

	0	1	2	3	4
0	0,0 3	0,1 1	0,2 5	0,3 2	0,4 1
1	1,0 4	1,1 0	1,2 -4	1,3 1	1,4 2
2	2,0 5	2,1 4	2,2 -5	2,3 3	2,4 1
3	3,0 8	3,1 2	3,2 -1	3,3 0	3,4 1

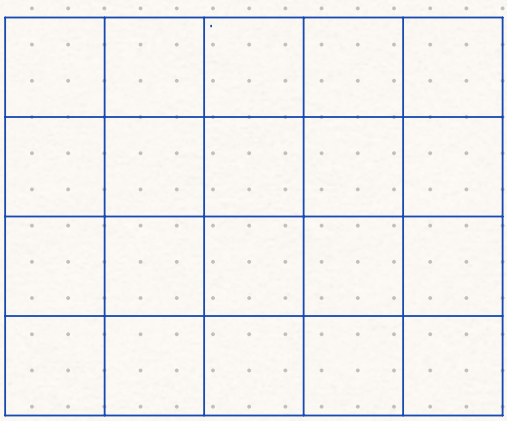
m

i = 0	i < m (4)	Body	i++
0	0 < 4 ✓		1

j=0	j < n (5)	Body	j++
0	0<5 ✓	A[0][0]	1
1	1<5 ✓	A[0][1]	2
2	2<5 ✓	A[0][2]	3
3	3<5 ✓	A[0][3]	4
4	4<5 ✓	A[0][4]	5
5	5<5 ✗	S.o.Println()	

i = 0	i < m (4)	Body	i++
1	1 < 4 ✓		2

j=0	j < n (5)	Body	j++
0	0<5 ✓	A[1][0]	1
1	1<5 ✓	A[1][1]	2
2	2<5 ✓	A[1][2]	3
3	3<5 ✓	A[1][3]	4
4	4<5 ✓	A[1][4]	5
5	5<5 ✗	S.o.Println()	



Sum of Rows

A	0	1	2	3	4
0	3 (0,0)	1 (0,1)	5 (0,2)	2 (0,3)	1 (0,4)
1	4 (1,0)	0 (1,1)	-4 (1,2)	1 (1,3)	2 (1,4)
2	5 (2,0)	4 (2,1)	-5 (2,2)	3 (2,3)	1 (2,4)
3	8 (3,0)	2 (3,1)	-1 (3,2)	0 (3,3)	1 (3,4)

row 0 : 3 + 1 + 5 + 2 + 1 = 12

row 1 : 4 + 0 + (-4) + 1 + 2 = 3

Output:

12
3
8
10

```
int m = A.length; int n = A[0].length
                  4
for (int i = 0; i < m; i++) {
    //find the sum of
      the row i
    int sum = 0
    for (int j=0; j<n; j++) {
        sum += A[i][j];
    }
    S.o.println(sum);
}
```

Sum of even :

A	0	1	2	3	4
0	0,0 3	0,1 1	0,2 5	0,3 2	0,4 1
1	1,0 4	1,1 0	1,2 -4	1,3 1	1,4 2
2	2,0 5	2,1 4	2,2 -5	2,3 3	2,4 1
3	3,0 8	3,1 2	3,2 -1	3,3 0	3,4 1

Find the sum of even elements in each row.

Output:

2
2
4
10