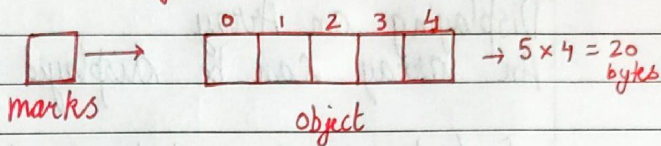


Chapter 6 - Arrays

Array is a collection of similar types of data

Use Case : Storing marks of 5 Students

`int[] marks = new int[5]` \Rightarrow `[data Type & Name;]`
 ↓ reference ↓ object



Accessing Array Elements

Array elements can be accessed as follows

marks[0] = 100

marks[1] = 70

\Rightarrow Note that index starts from 0

marks [4] = 98

So in a nut shell, this is how array works:

```
17 int [] marks;
```

→ Declaration !

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

→ Memory Allocation!

27 `int l marks = new int[5];` → Declaration + Memory Allocation!

32. `int[] marks = {100, 70, 80, 71, 98}` → Declare + Initialize!

Array indices starts from 0 and goes till $(n-1)$ where n is the size of the array.

Array length

Arrays have a length property which gives the length of the array

marks.length \rightarrow gives 5 if marks is a reference to array with 5 elements

Displaying an Array

An array can be displayed using a for loop:

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

```
{
```

```
    Sout (marks[i]);
```

```
}
```

\Rightarrow Array Traversal

Quick Quiz: Write a Java program to print the elements of an array in reverse order.

For-each loop in Java

Array elements can also be traversed as follows:

```
for (int element : Arr) {
```

```
    Sout (element);
```

```
}
```

\Rightarrow Prints all the elements

Multidimensional Arrays

Multidimensional Arrays are Array of Arrays

Each element of a M-D array is an array itself
marks in the previous example was a 1-D array.

Multidimensional 2-D Array

A 2-D array can be created as follows:

```
int [][] flats = new int [2][3]
```

↳ A 2-D array of 2 rows + 3 columns

We can add elements to this array as follows

flats [0][0] = 100

flats [0][1] = 101

flats [0][2] = 102

⋮

& so on!

This 2-D array can be visualised as follows:

	[0]	[1]	[2]
	Col 1	Col 2	Col 3
[0] Row 1	(0, 0)	(0, 1)	(0, 2)
[1] Row 2	(1, 0)	(1, 1)	(1, 2)

Similarly a 3-D array can be created as follows:

```
String [][][ ] arr = new String [2][3][4]
```