Introduction to Array and Strings

made by vijay singh

Introduction to Arrays in Java

1.What is an Array?

An **array** in Java is a container that stores **multiple values of the same data type** in one variable.

Example:
If you want to store marks of 5 students, instead of writing:

```
int m1 = 10;
int m2 = 20;
int m3 = 30;
```

You can use an array:

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

Now one variable stores 5 values.

2.Declaration of Array in Java

Declaration means telling Java:

- What type of data?
- What is the name?

Syntax:

```
dataType[] arrayName;
```

Example:

```
int[] arr;
```

This only declares the array.
Memory is not created yet.

3.Creating Array (Memory Allocation)

In Java, we use **new keyword**.

int[] arr = new int[5];

This creates space for 5 integers.

Important:

- Index starts from **0**
- Last index = size - 1

So here:

- arr[0]
- arr[1]
- arr[2]
- arr[3]
- arr[4]

4.Initialization of Array

Method 1: Store values one by one

arr[0] = 10;
arr[1] = 20;
arr[2] = 30;

Method 2: Direct initialization

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

Java automatically counts size.

5.Traversing an Array (Accessing Elements)

Traversal means printing or accessing each element.

Using for loop

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

for(int i = 0; i < arr.length; i++) {
   System.out.println(arr[i]);
}
```

👉 **arr.length gives the size of array.**

Important Points About Java Arrays

Index starts from 0
Size is fixed after creation
All elements must be same type
length is used to find size
If not initialized, default values are stored

Default values:

- int → 0
- double → 0.0
- boolean → false
- String → null

Types of Arrays in Java

1.1D Array

int[] arr = new int[5];

2.2D Array (Matrix)

int[][] matrix = new int[3][3];

Example:

```
int[][] matrix = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9}
};
```

Introduction to String in Java

In Java, **String is not a simple array**.
It is a **class**.

Java provides a built-in class:

String

Creating a String in Java

Method 1 (Most common)

String name = "Vijay";

Method 2

String name = new String("Vijay");

Important String Methods

1.Length

name.length();

2.Access character

name.charAt(0);

3.Compare Strings

name.equals("Vijay");

Don’t use == for comparing strings.

4.Convert to Uppercase

name.toUpperCase();

5.Concatenation (Join)

String a = "Hello";
String b = "World";

String c = a + " " + b;

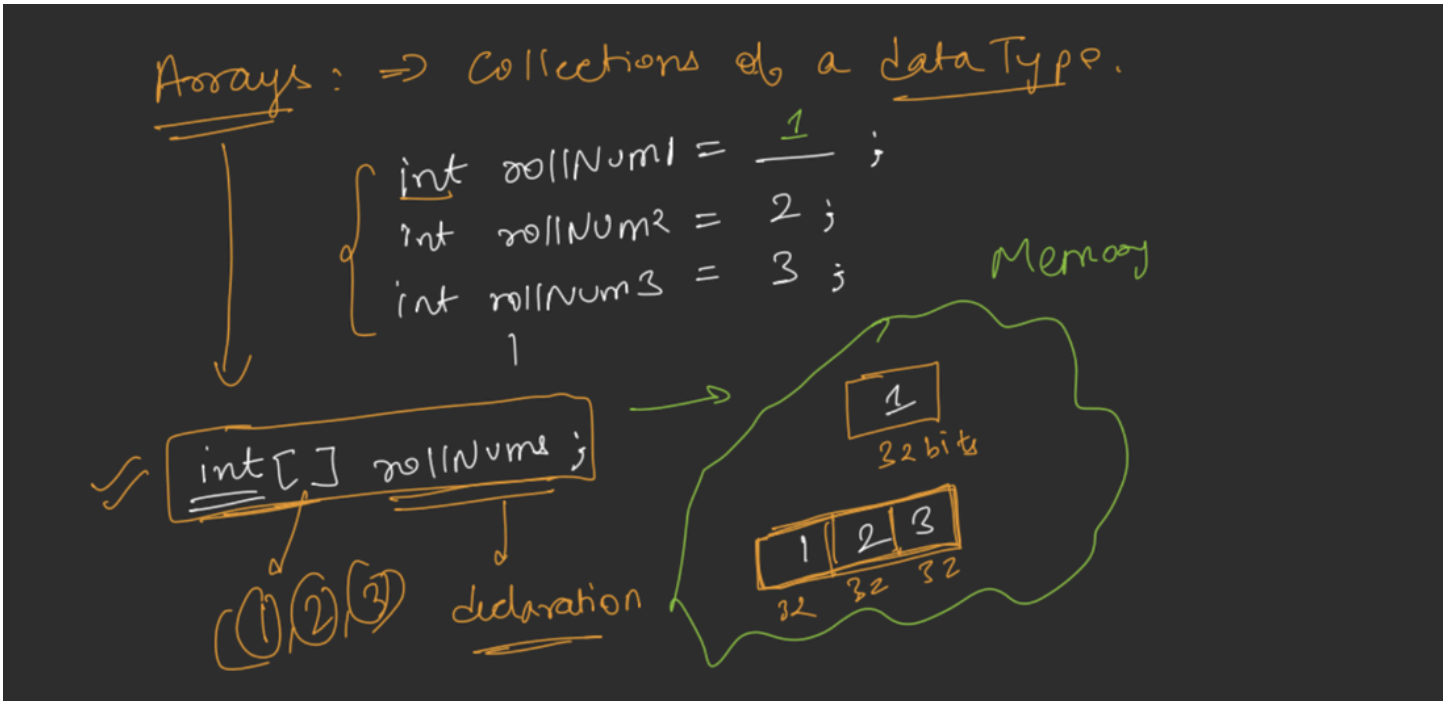
Traversing a String

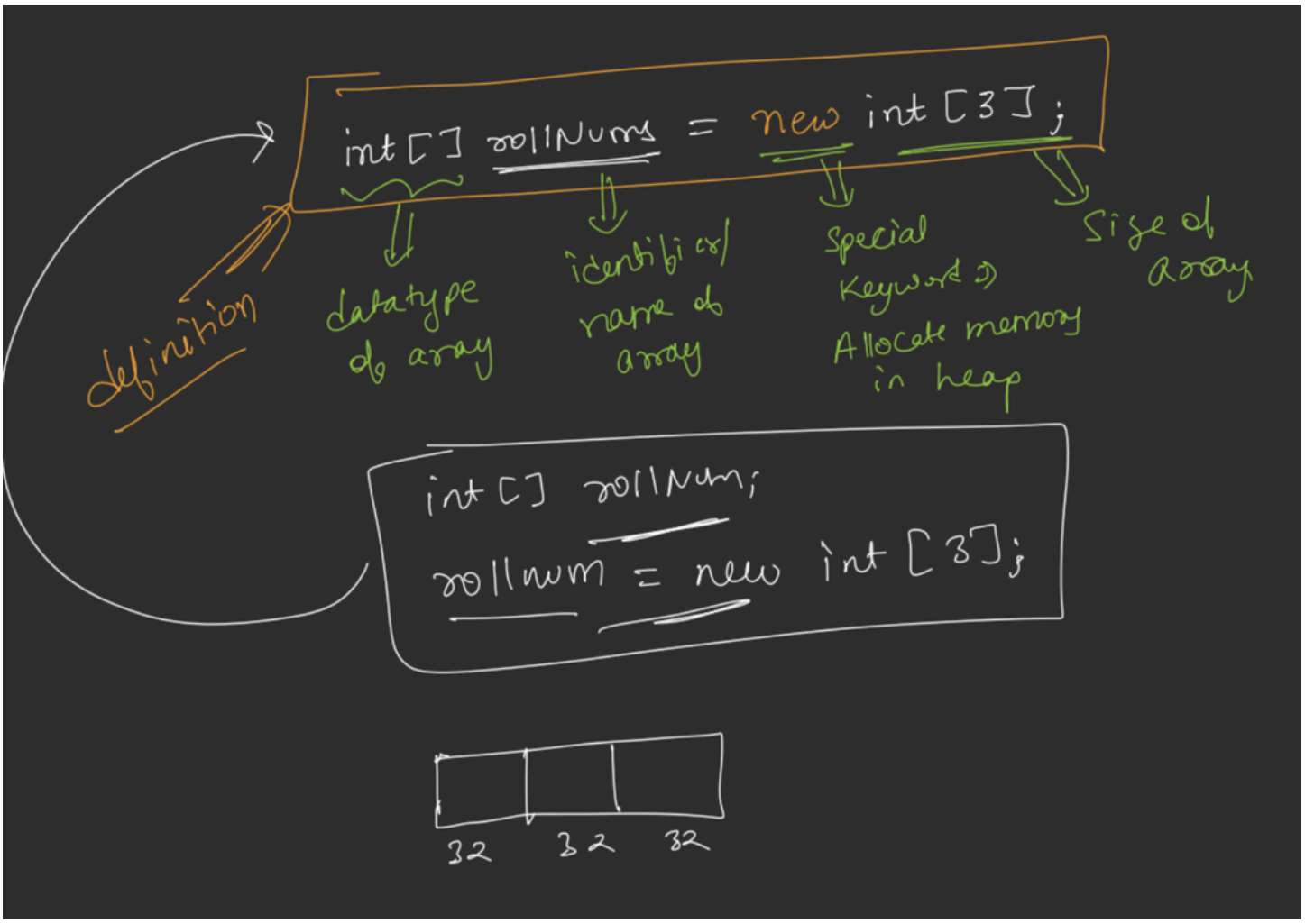
String name = "Vijay";

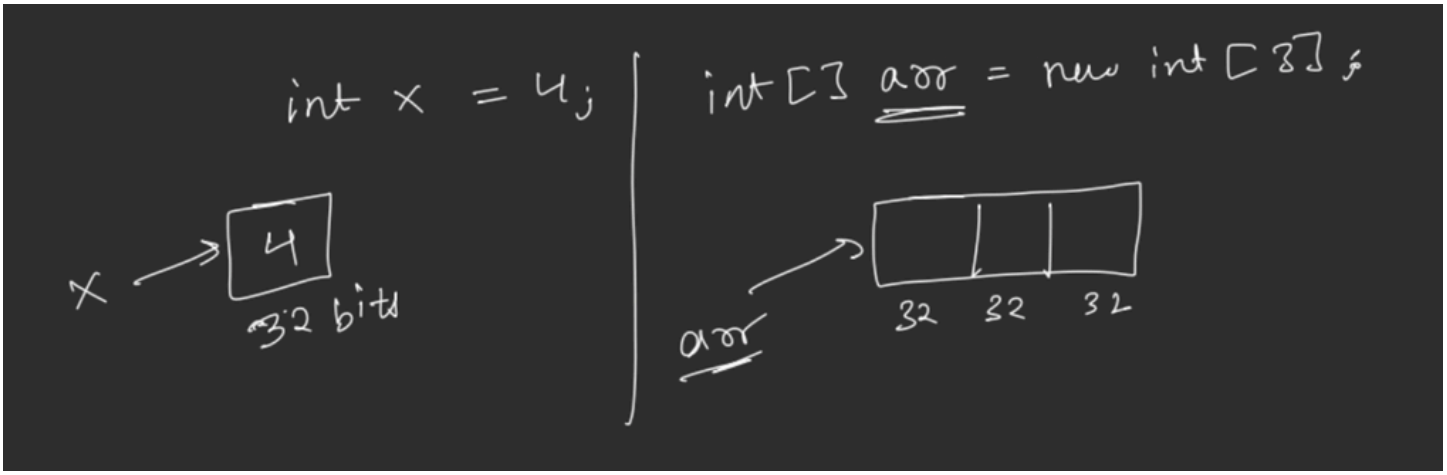
for(int i = 0; i < name.length(); i++) {
  System.out.println(name.charAt(i));
}

Difference: Array vs String in Java

Array	String
Stores same type values	Stores characters
Fixed size	Cannot change (immutable)
Uses []	Uses String class
length property	length() method




int [] rollNums = new int [3];
definition
datatype of array
identifier/ name of array
Special Keyword => Allocate memory in heap
Size of array
int [] rollNum;
rollnum = new int [3];
32
32
32


int x = 4;
x
4
32 bits
int [] arr = new int [3];
arr
32
32
32

r1 → 101
r2 → 102
r3 → 103

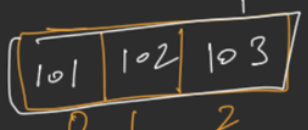
int [] rollNums = new int [3];

rollNums[0] = 101;
rollNums[1] = 102;
rollNums[2] = 103;

System.out.println(rollNums[0]); // 101
(rollNums[1]); // 102
(rollNums[2]); // 103

int [] rollNums = new int [3];

i
i=3



x = 101 ~~102~~ ~~103~~ 104

101
102
102

int x = 101;

```
for(int i=0; i<3; i++){
    rollNums[i] = x;
    x++;
}
```

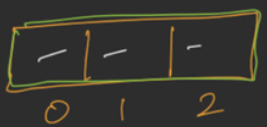
```
for(int i=0; i<3; i++){
    System.out.println(rollNums[i]);
}
```

101
102
103

Multi-Dimensional Array :

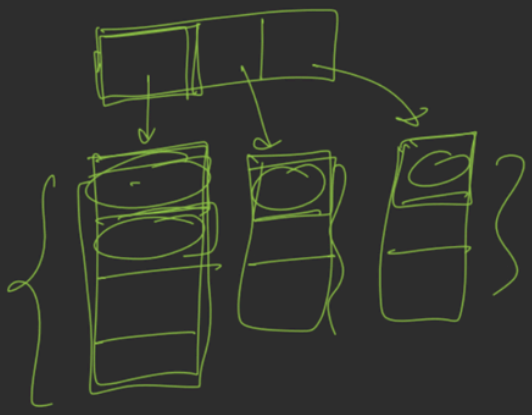
1-D
↳ Array of int

```
int [] rollNums = new int [3];
```

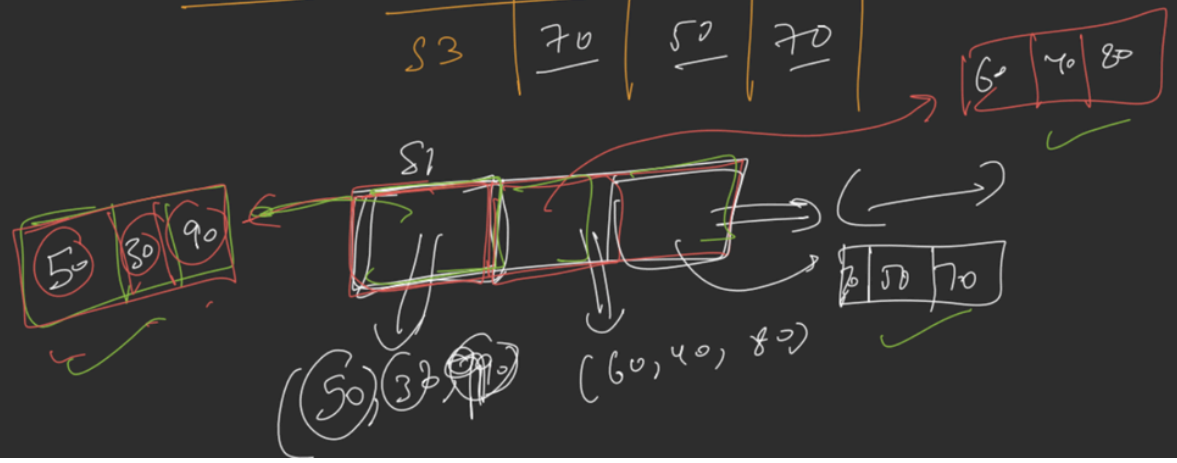


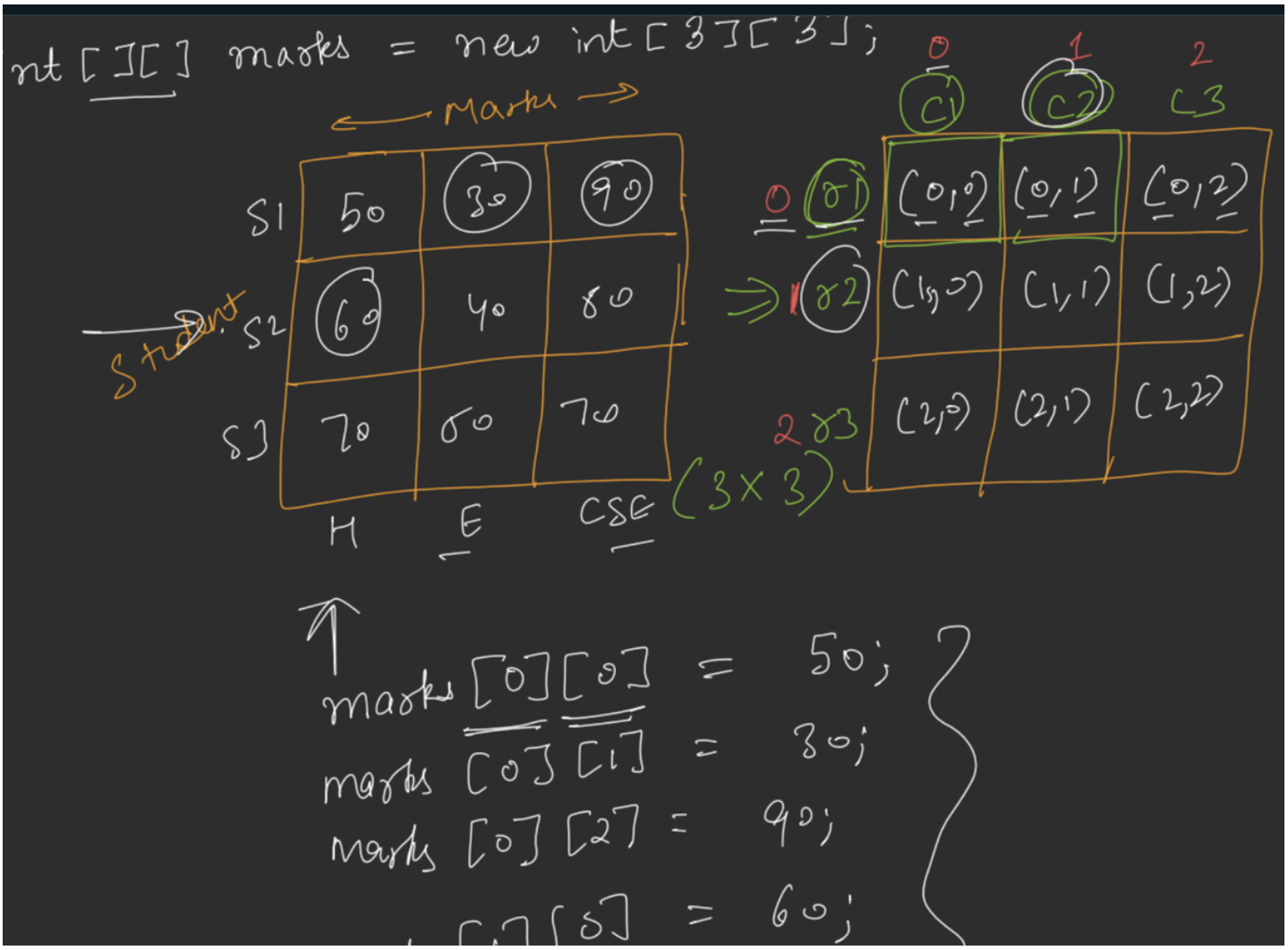
1-D Array

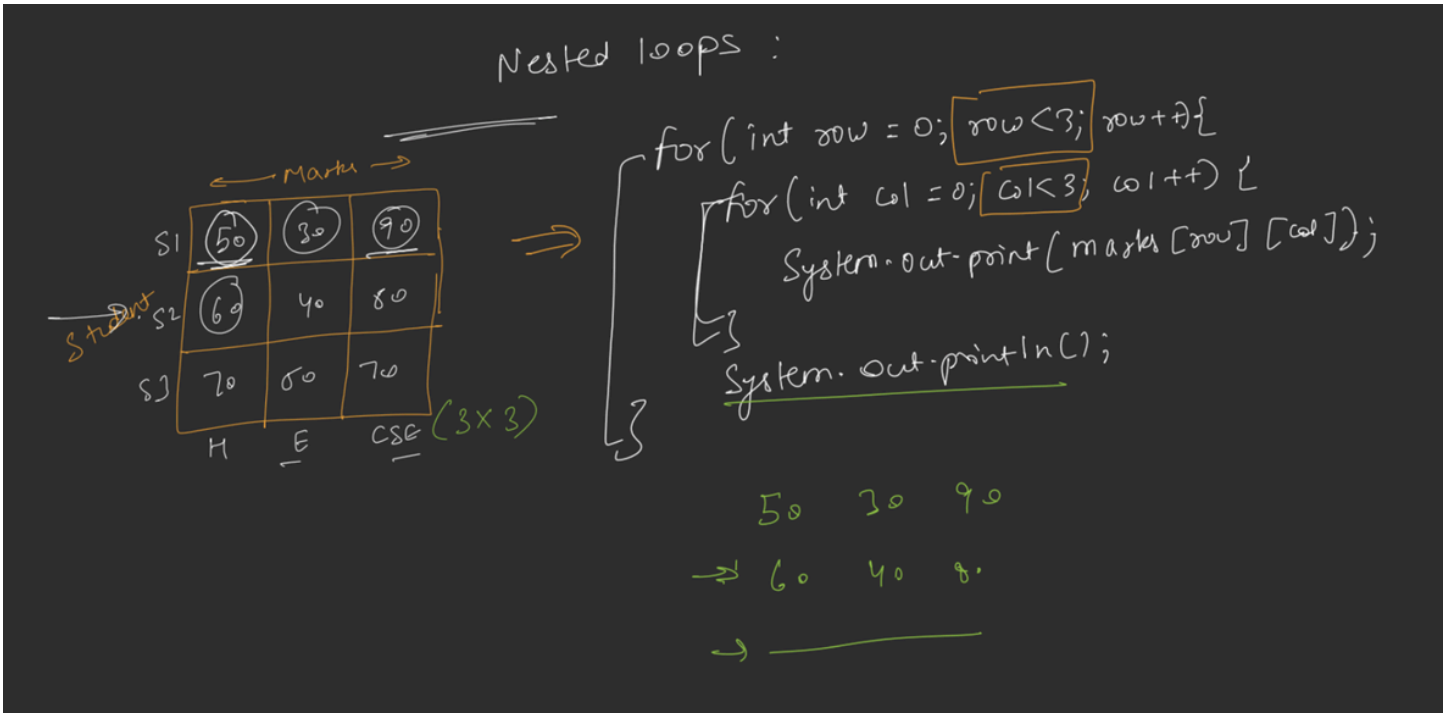
2-D Array :
↳ Array of Arrays



Students	Hindi	English	CSE
S1	50	30	90
S2	60	40	80
S3	70	50	70

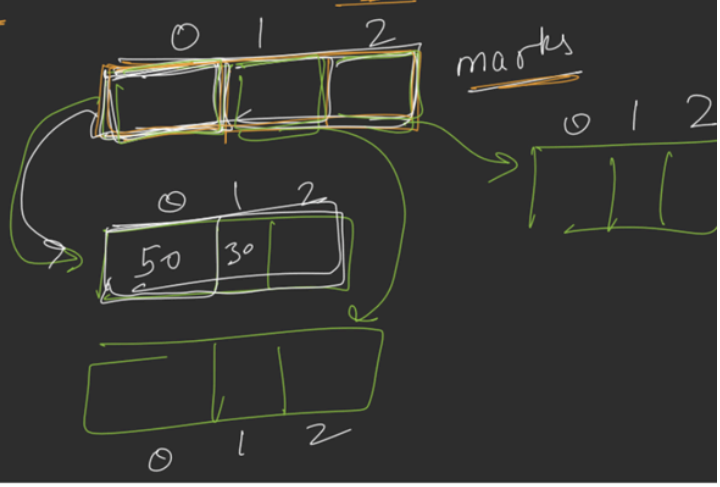



nt [][] marks = new int[3][3];
Marks
S1 50 30 90
S2 60 40 80
S3 70 50 70
H E CSE
Student
(3×3)
0 1 2
C1 C2 C3
0 (0,0) (0,1) (0,2)
1 (1,0) (1,1) (1,2)
2 (2,0) (2,1) (2,2)
marks[0][0] = 50;
marks[0][1] = 30;
marks[0][2] = 90;
[1][0] = 60;


Nested loops :
Marks
S1 50 30 90
S2 60 40 80
S3 70 50 70
H E CSE
Student
(3×3)
for(int row = 0; row<3; row++){
for(int col = 0; col<3; col++){
System.out.print(marks[row][col]);
}
System.out.println();
}
50 30 90
60 40 80

Conceptual Representation of 2-D Arrays

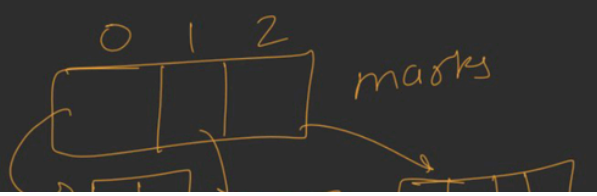
```
int [][] marks = new int [3][3];
marks[0][0] = 50;
[0][1] = 30;
```

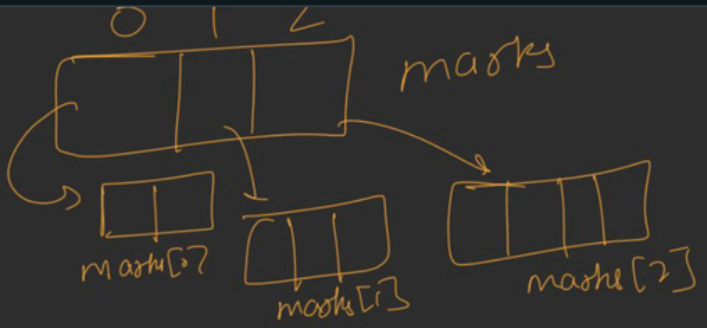



```
for(int row = 0; row<3; row++){
    for(int col = 0; col<3; col++){
        System.out.print(marks[row][col]);
    }
    System.out.println();
}

for(int row = 0; row < marks.length; row++){
    for(int col = 0; col < marks[row].length; col++){

    }
}
```

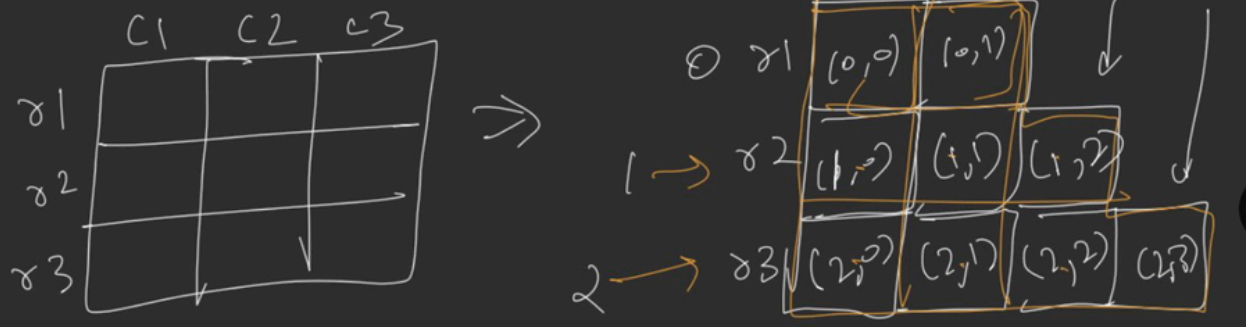
Student	H	E	CSE
S1	50	30	90
S2	60	40	80
S3	70	50	70

(3×3)





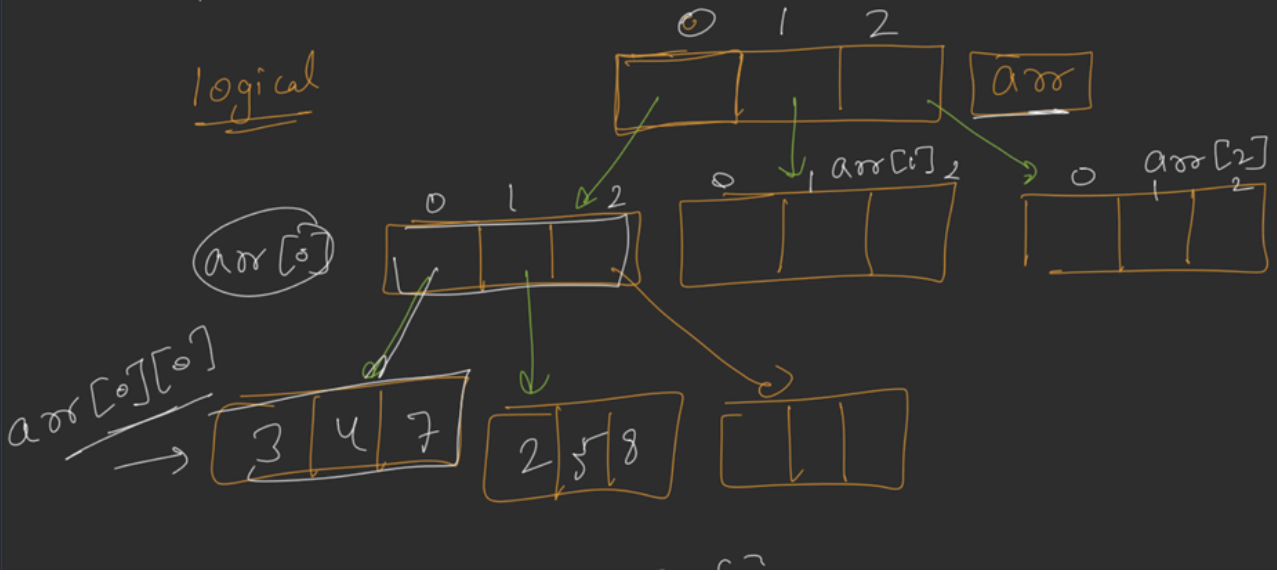

```
int[][] marks = new int[3][ ];
int marks[][] = new int[3][7];
→ marks[0] = new int[2];
→ marks[1] = new int[3];
→ marks[2] = new int[4];
```

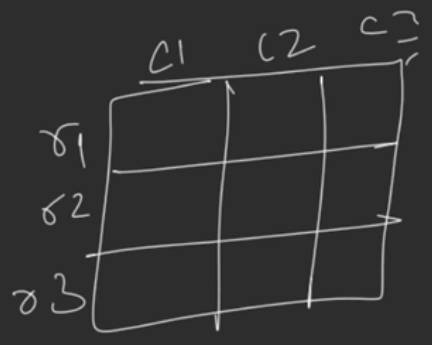


3-D Array :

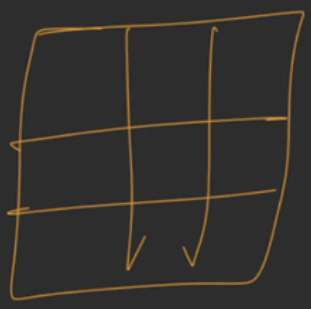
```
int[][][] arr = new int[2][3][3];
```

logical





2D



2D

2D

4D, 5D, ---------- ND

Multi-Dimensional:

Multi-Dimensional:

1-D

```
int[] rollNums = new int[3];
```

```
rollNums[0] = 4;
rollNums[1] = 5;
```

```
int[] rollNums = {4, 5, 6};
```