brush[0].ChangeSize(thin);

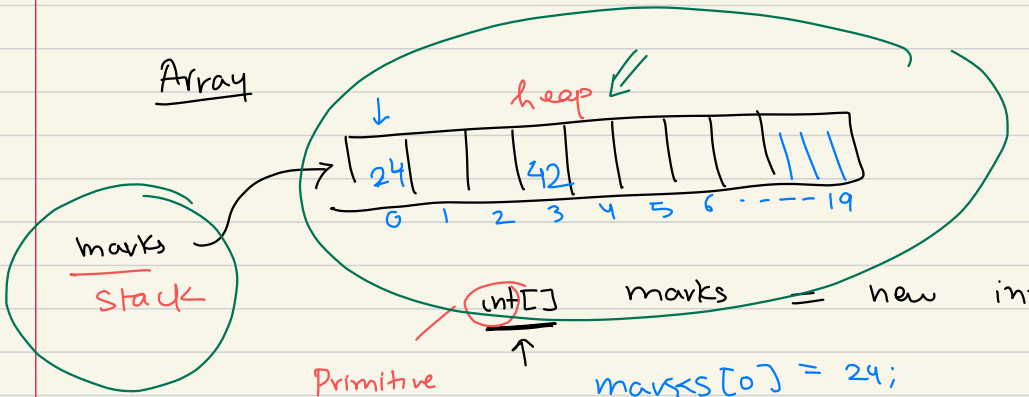
Topics

1. "2D Arrays" — 80%
2. "Array Lists" — 20%

+

✓ → Functions PPT] Today
✓ → Maths Code]
→ Extra Questions] Tomorrow

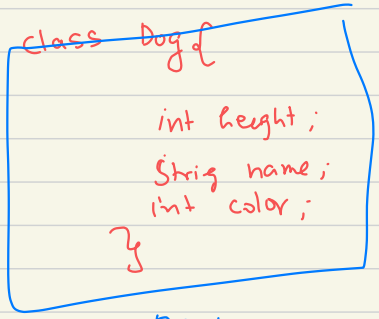
Array



- ✓ Create
- ✓ Update
- ✓ Delete
- ✓ Print

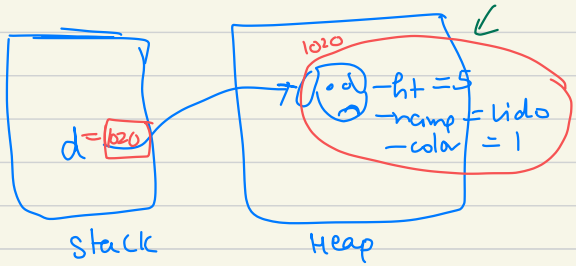
```
marks[0] = 24;
marks[3] = sc.nextInt();
```

```
char[] a, = ---
boolean[] b, = ---
float[] f = ---
```



Dog objects

Dog d1 = new Dog();



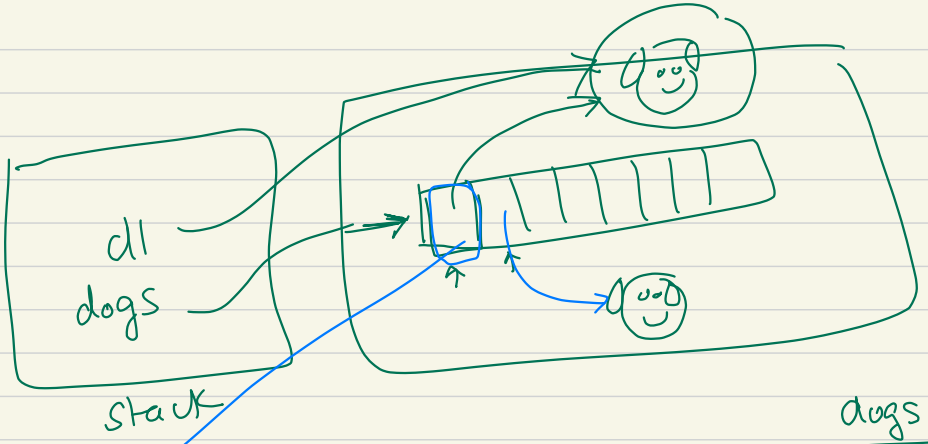
d.height = 5
d.name = "Lido"
d.color = 1

Array of objects ⇒ Dog[] dogs = new Dog[100];

Dog[] — Array; dogs — Name

↘ creates an array of 100 dogs

TV[] tvs = new TV[100];



Each bucket of array is obj ref.

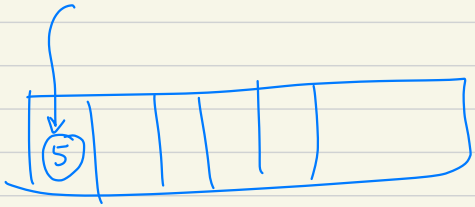
dogs[0] = dl;

~~dogs[0].height = 5~~

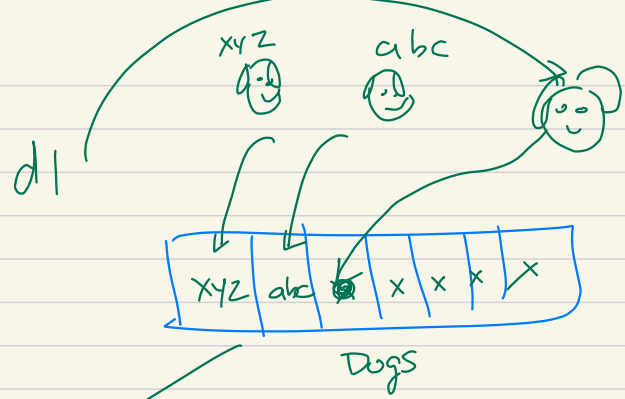
Address

dogs[i] = new Dog();

in any part of heap



marks[0] = 5



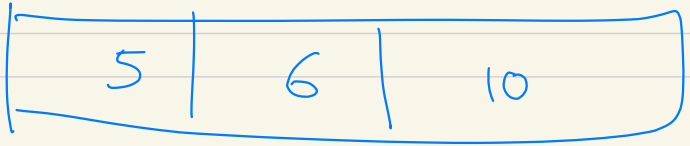
dogs[0] = new Dog()

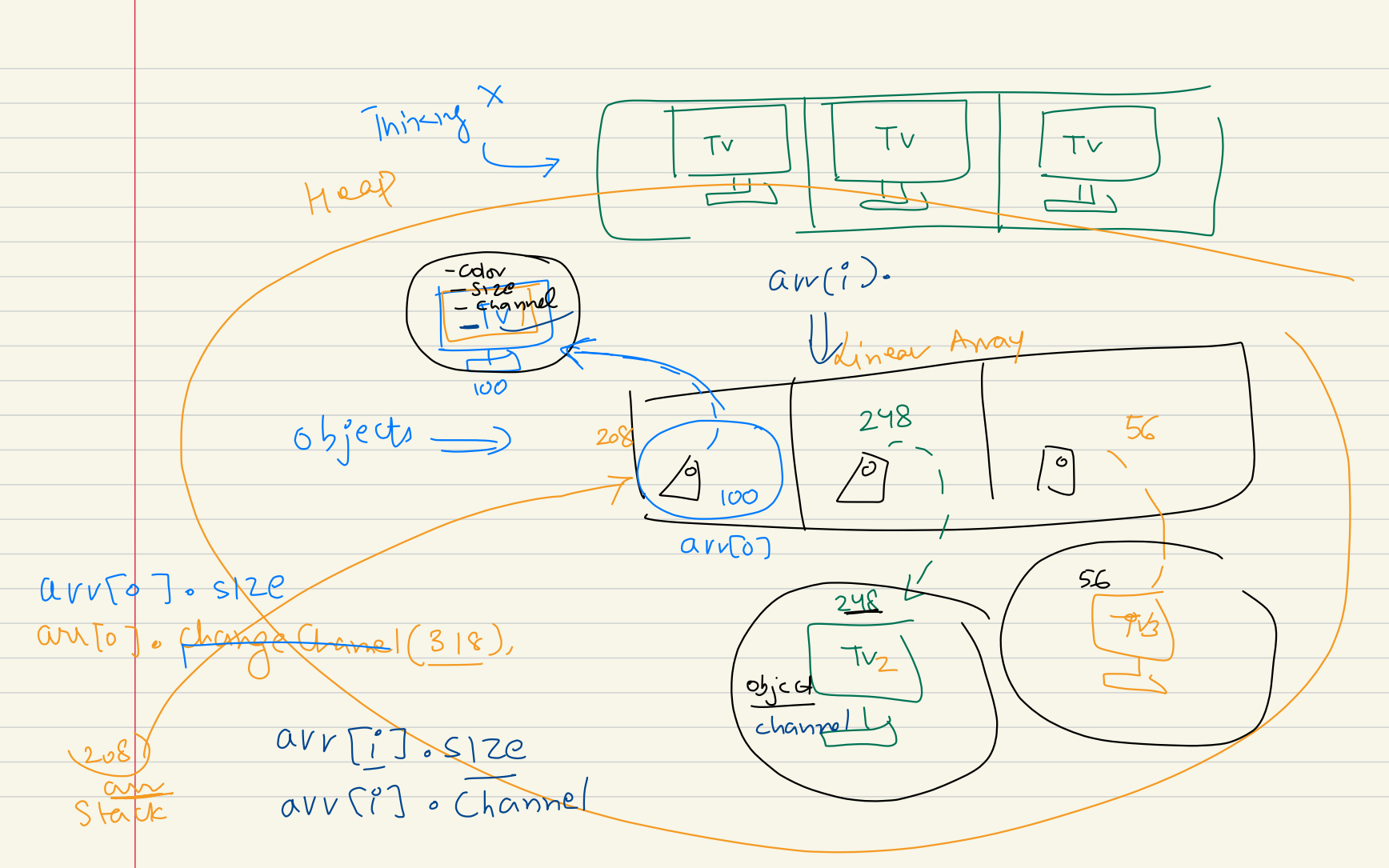
dogs[1] = new Dog()

dogs[2] = d1;

Each bucket of the array is inside heap but it only store a reference to Dog object

int ⇒

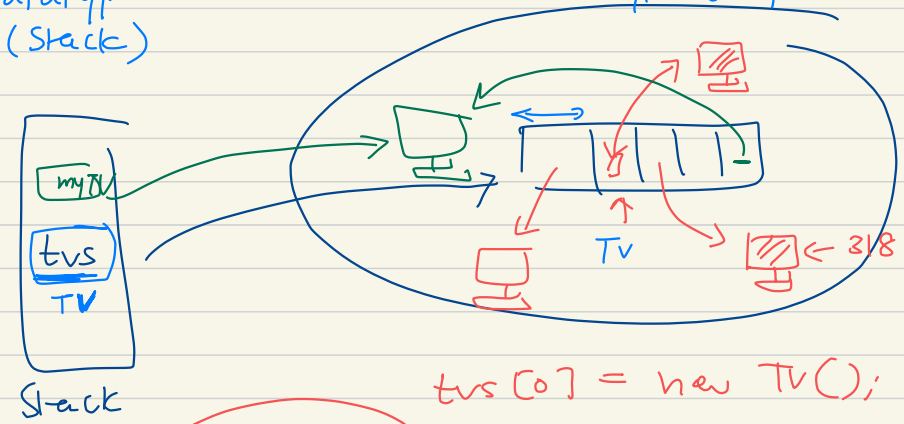



Thinking
Heap
Tv
Tv
Tv
-color
-size
-channel
TV
100
arr(i).
Linear Array
objects
208
100
248
56
arr[0]
arr[0].size
arr[0].changeChannel(318);
248
Tv2
objects
channel
56
Tv3
208
arr
Stack
arr[i].size
arr[i].channel

```
Tv[]  tvs  =  new  Tv[5];
```

Datatype (Stack)

Datatype (Heap)



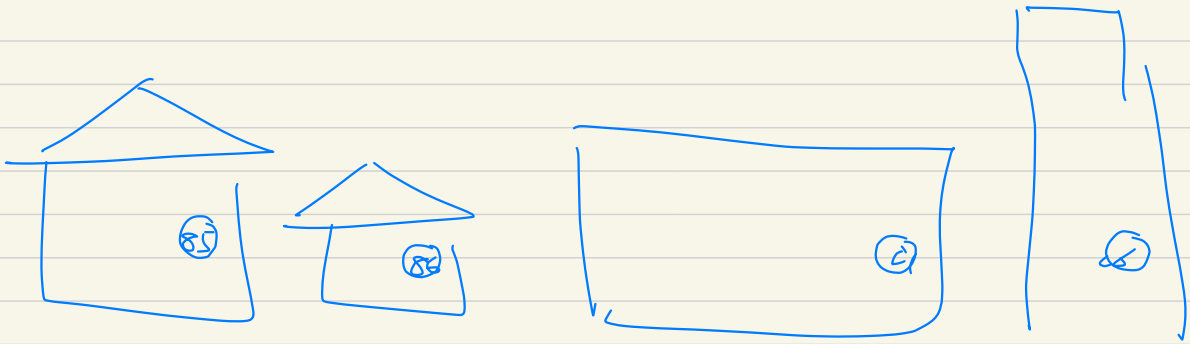
TV
myTV = new TV()

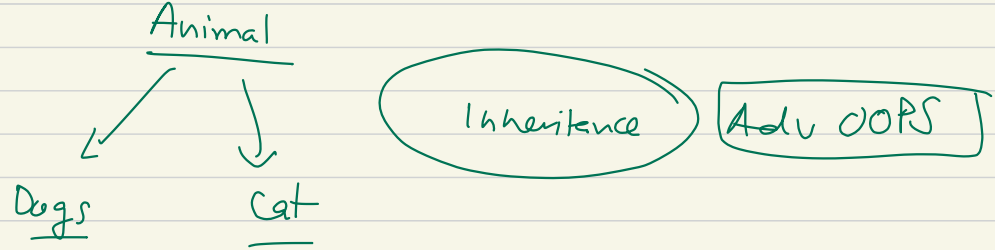
tvs[0] = new Tv();

tvs[1] = new Tv();

tvs[2] = new Tv();
tvs[2].channel = 318;

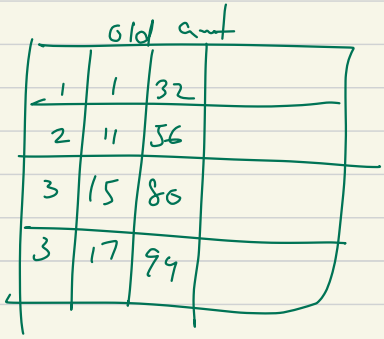
tvs[3] = myTV;



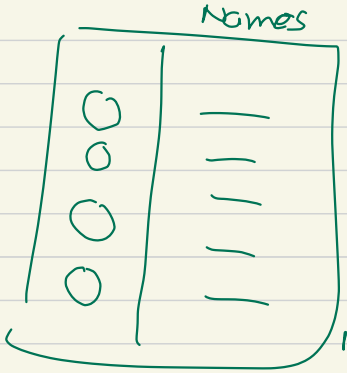


Cats → Dogs X
Dogs → Cats X
Cats → Animal
Dog → Animal

2D Arrays

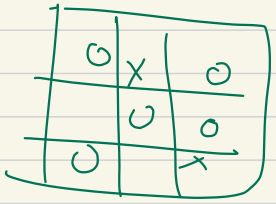


excel / sales data



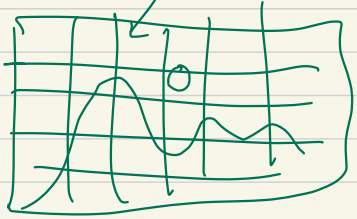
N, 2 ds

EVM



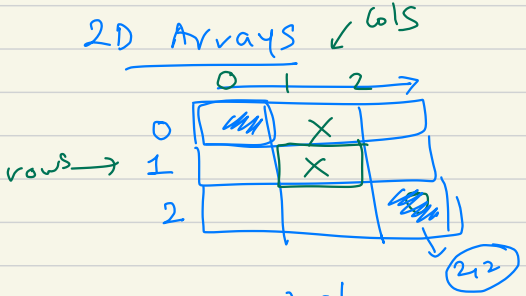
Game

State of game



Matrix (SD card)

2D Arrays



X O

```
mat[1][1] = 'x'
mat[2][2] = '0'
mat[0][1] = 'x';
```

mat[row][col] → Read it / update

Code

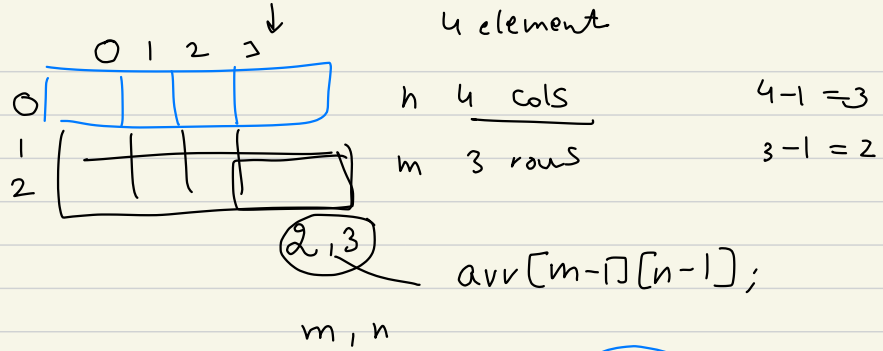
m, n

⇒ first-cell = (0,0)

⇒ last-cell = (m-1, n-1);

```
int[][] mat = new int[3][3];
char[][] mat = new char[rows][cols];
```

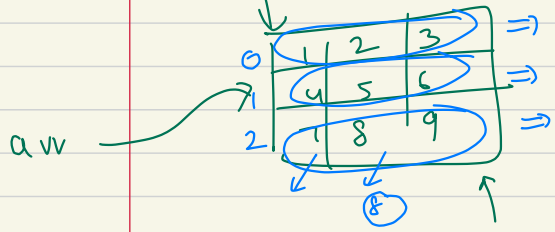
(Rows, cols)



2D Arrays ⇒ Array of Arrays (obj)

↳ Array of Object

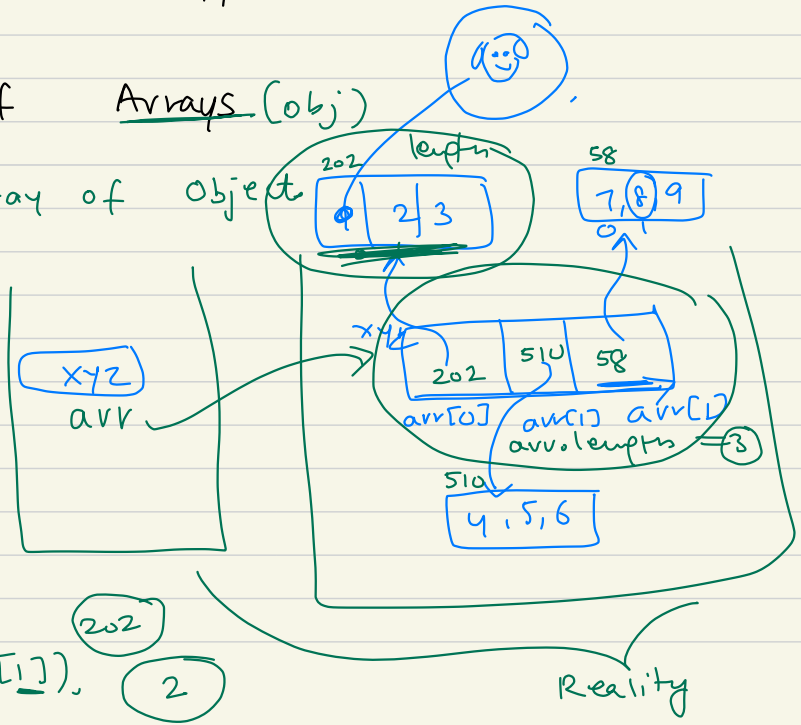
Visualise



arr[0] = 1,2,3

print (arr[0]); 202

print (arr[0][1]); 2

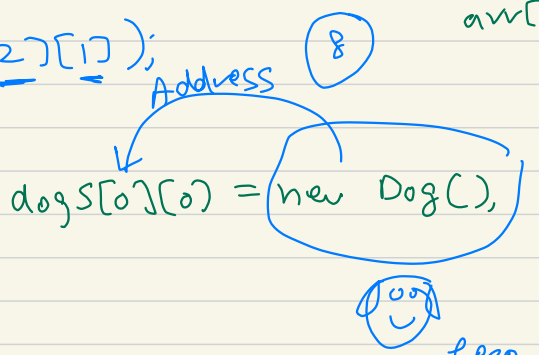


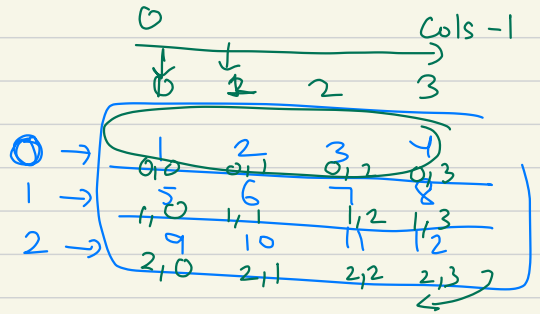
print (arr);

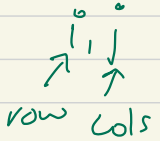
print (arr[2][1]);

arr.length
arr[0].length.

2D arrays of dogs



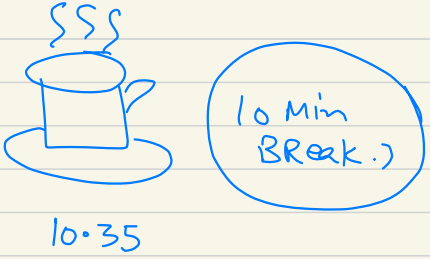




0 1

0, 1

arr[0][0] arr[0][1]



10:35

DIY

I

	0	1	2	3
0	1	2	3	4
1	5	6	10	15
2	26	32	48	64

N x M

Searching

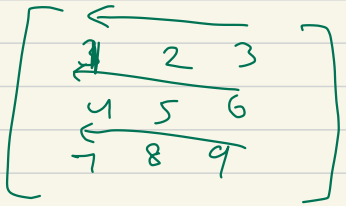
Find x, y

of (Key)

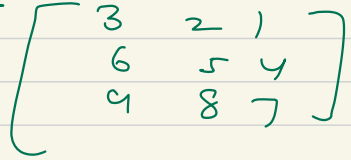
Input = 32

output = 2, 1

II

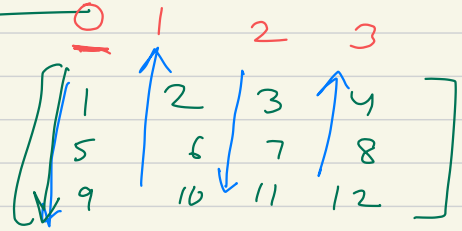


Mirror



III

Bonus:



for (col = 0 ——— last - col)

~~even~~ ~~odd~~ ↓

odd ↑

Printed 1, 5, 9, 10, 6, 2 3, 7, 11, 12, 8, 4

Code it

DIY

75%

~~41%~~

Now

25 %

~~59%~~

Next class

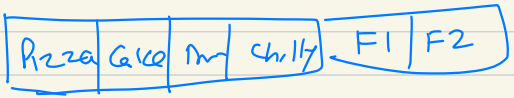
Strings

OOPS

→ Tanvi

prateek.narang@scaler.com

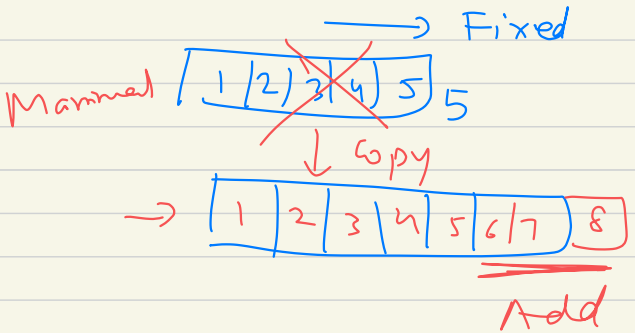
grow your storage →

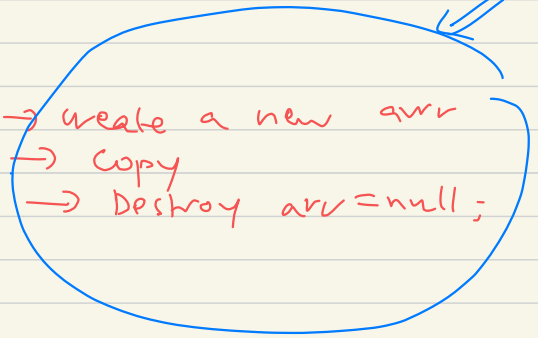


Java ⇒ Collection framework

↓

ARRAYList





Dynamic Array → can grow when capacity limit grows.