Doubts

1. Remove / Insert in the Array → Insert an element at X
   → Delete an element at specified pos. X

2. How many arguments to pass ?

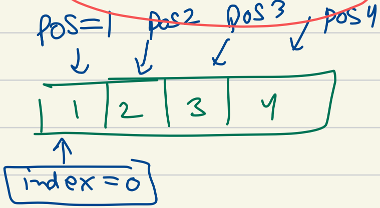
Topic → 3. Dynamic size Arrays → Yes.

4. ARRAY out of Bounds.

5. Return stmt (non-void) / Return arrays from a fn

6. Position (laymen term)

   Index



pos = 1+idx;

7. int [] arr vs int arr [] same

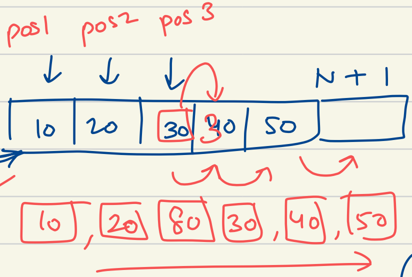
8 $3.14 * A * A$ ✓ Math.PI * A x A

I N = 5

10, 20, 30, 40, 50

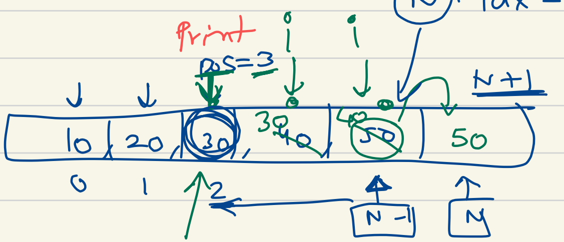
↳ x = 3

y = 80



how to shift

N idx = pos-1;

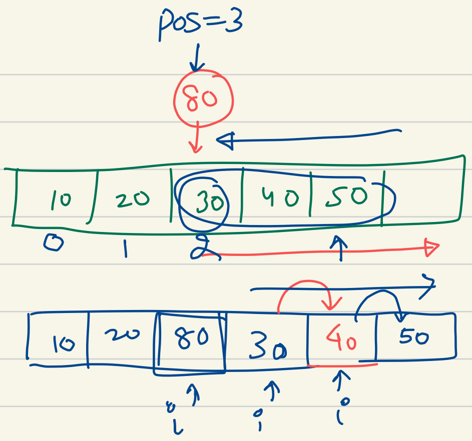


i=N-1

Print

```
for (i= N-1 ; i >= pos-1, i--){
        a[i+1] = a[i];
}
a[pos-1] = y;
```

idx= pos-1



Insert
↓
N+1

Copying

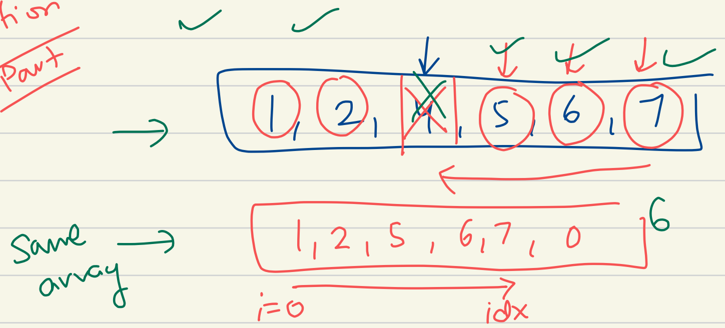
```
a[i+1] = a[i]
i--;
```

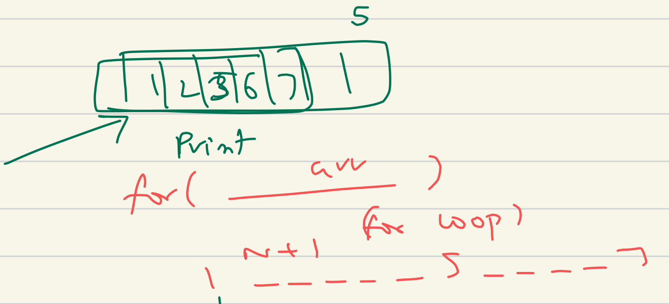
Outside loop

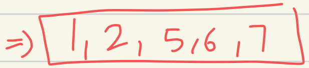
```
A[i] = Y;      //80
```

Deletion

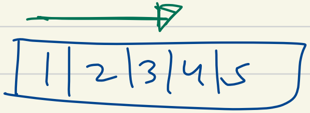
~~Part~~







What you want from this fn.



② printArray (int [ ] arr, ~~int n~~) {

int n = arr.length;

}

Needed for logic

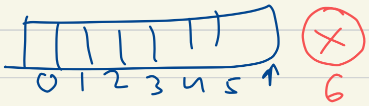
Parameters (definition)

printArr (arr, 6) → Argument (values) → values sent to parameters.

calling

Array Out Bounds

n = 6



stop at the last idx.

```
for (i=0; i ~~<=~~ < n, i++
    arr[i]
```

outside the memory region

Q

shopping_list ⇒ (100, 500, 700, 200)

i=0 i=1 i=2 i=3

Add

5, 1, 2, 3, 4, 5

Traverse

Q

Array (Repeating Elements)

0	1	2	3	4	5	6	7
5	6	7	2	7	3	7	4

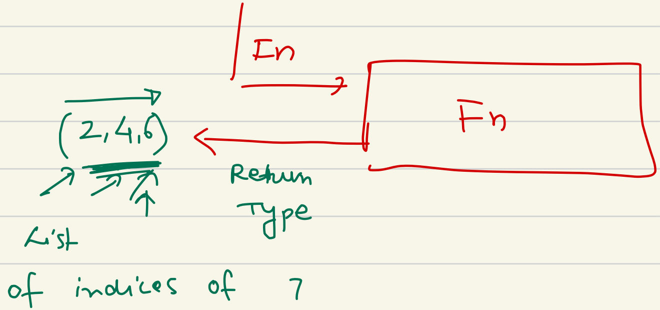
7

Return

All occurences (indices) of a given element = 7

Think...

"9.52"

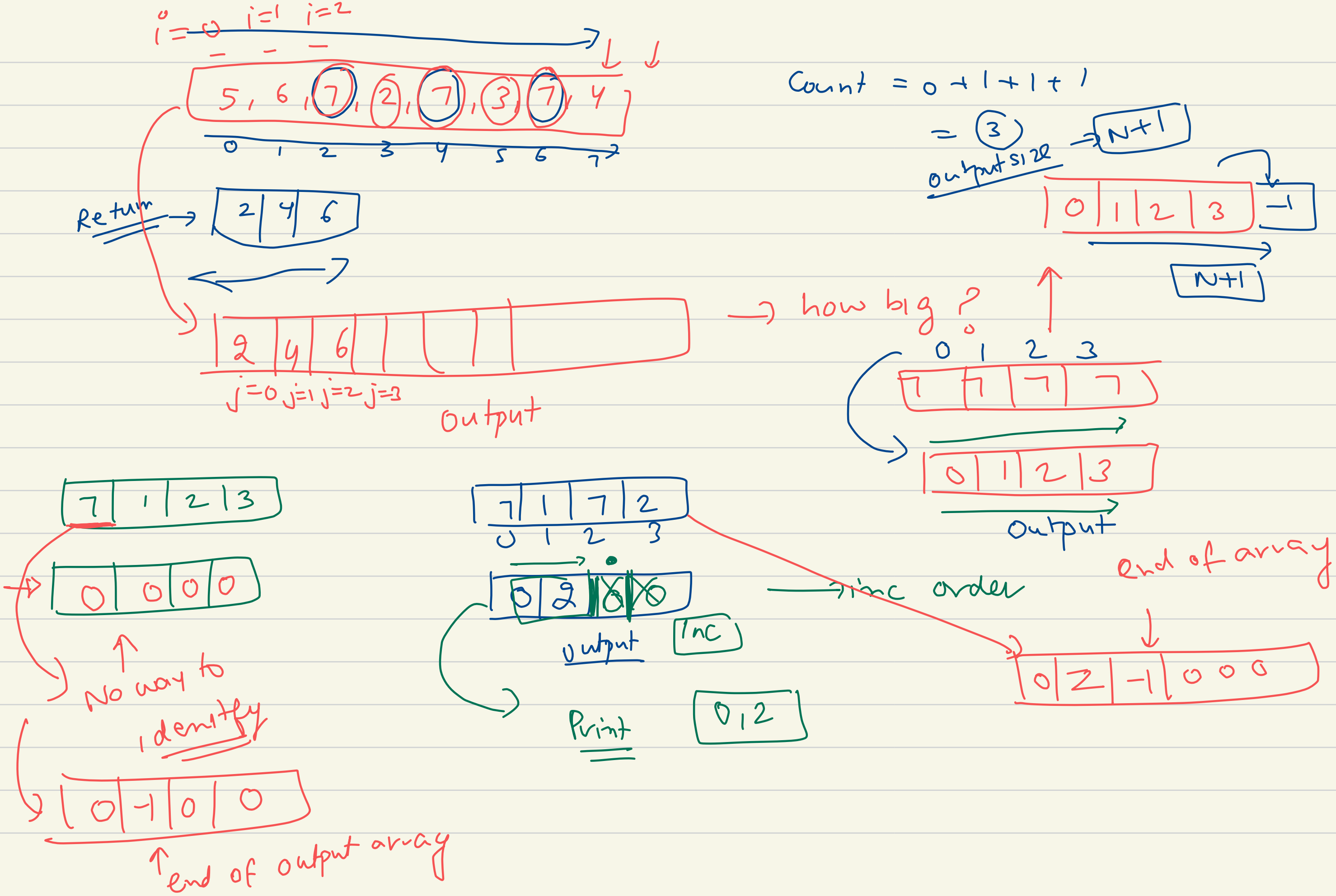


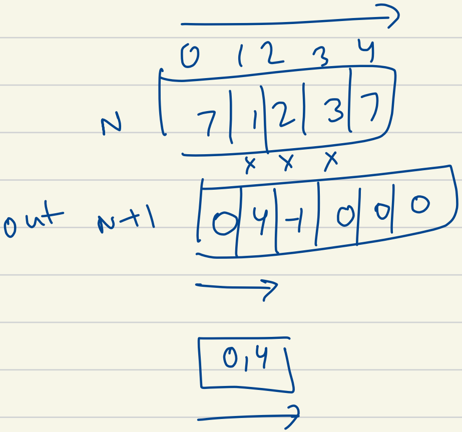
public static ___ searchAll(int[] a)

{

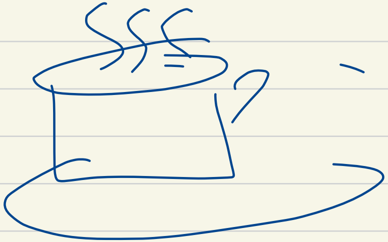
}

3


i=0 i=1 i=2
5, 6, 7, 2, 7, 3, 7, 4
0 1 2 3 4 5 6 7
Return
2 | 4 | 6
2 | 4 | 6
j=0 j=1 j=2 j=3
output
Count = 0 + 1 + 1 + 1
= 3
output size
N+1
0 | 1 | 2 | 3 | -1
N+1
how big ?
0 1 2 3
7 | 7 | 7 | 7
0 | 1 | 2 | 3
Output
7 | 1 | 2 | 3
0 | 0 | 0 | 0
No way to identify
0 | -1 | 0 | 0
end of output array
7 | 1 | 7 | 2
0 1 2 3
0 | 2 | 0 | 0
output
Inc
Print
0, 2
inc order
end of array
0 | 2 | -1 | 0 0 0


0 1 2 3 4
N
7 1 2 3 7
x x x
out
N+1
0 4 -1 0 0 0
0,4

Part - II



– "1/0030"

↳ ARRAY LISTS → Behaviour is pre-coded ⇒ Collections Framework

↓

(Lot of Data Structures)

↳ Wrapper Classes

↳ Problem ☆ (Nested loops on Arrays)

0	1	2	3	4	5	6
1	2	3	7	7	-	7

↓

Fixed size

3	4	6

Dynamic

→

grow

ARRAY LIST

↳ idea

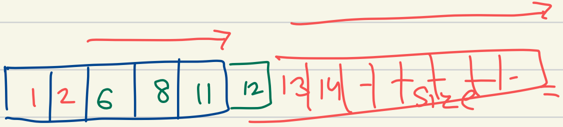
↳ use it

ArrayList – it is an array that can grow in size based upon requirement

Degrades the performance

copying data is ~~CPU~~ time taking

Double itself (in a new memory) whenever the capacity gets full



Capacity = ~~5~~ ~~6~~

↓

Double Capacity = 10

add(6) ✓

add(8) ✓

add(11) ✓

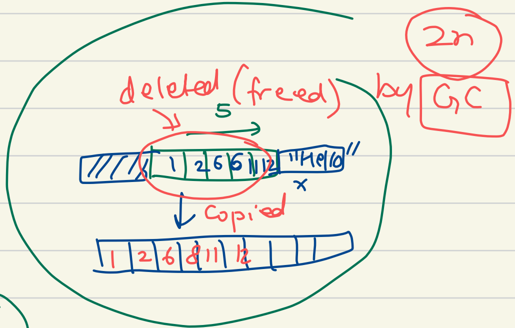
add(12) ↓ Doubling

add(13) ✓

add(14) ✓

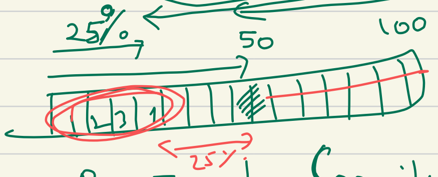
fast

Init ⇒ ArrayList capacity · n



Inefficient

26th element



Req = $\frac{1}{4}$ Capacity

Java Docs

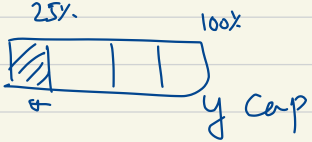
Reduce

" Scheme "

Rest 2 Rohis 4 ~~code~~ Rohis

	Capacity	Size	1 Happens?	
Insert ①	X	X	Doubling	X ↷ 2X Cap

	Capacity	Size		
Remove	y	y/4	Reduce	y → y/2 Reduction

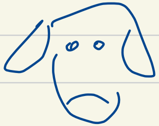


Collections

↓

Objects

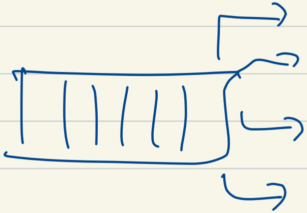
Object



Name
Height
color
} Knows

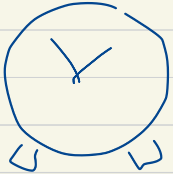
eat()
bark()
run()
} actions

dog.run()



add()
remove()
get()
↳ set()

ArrayList



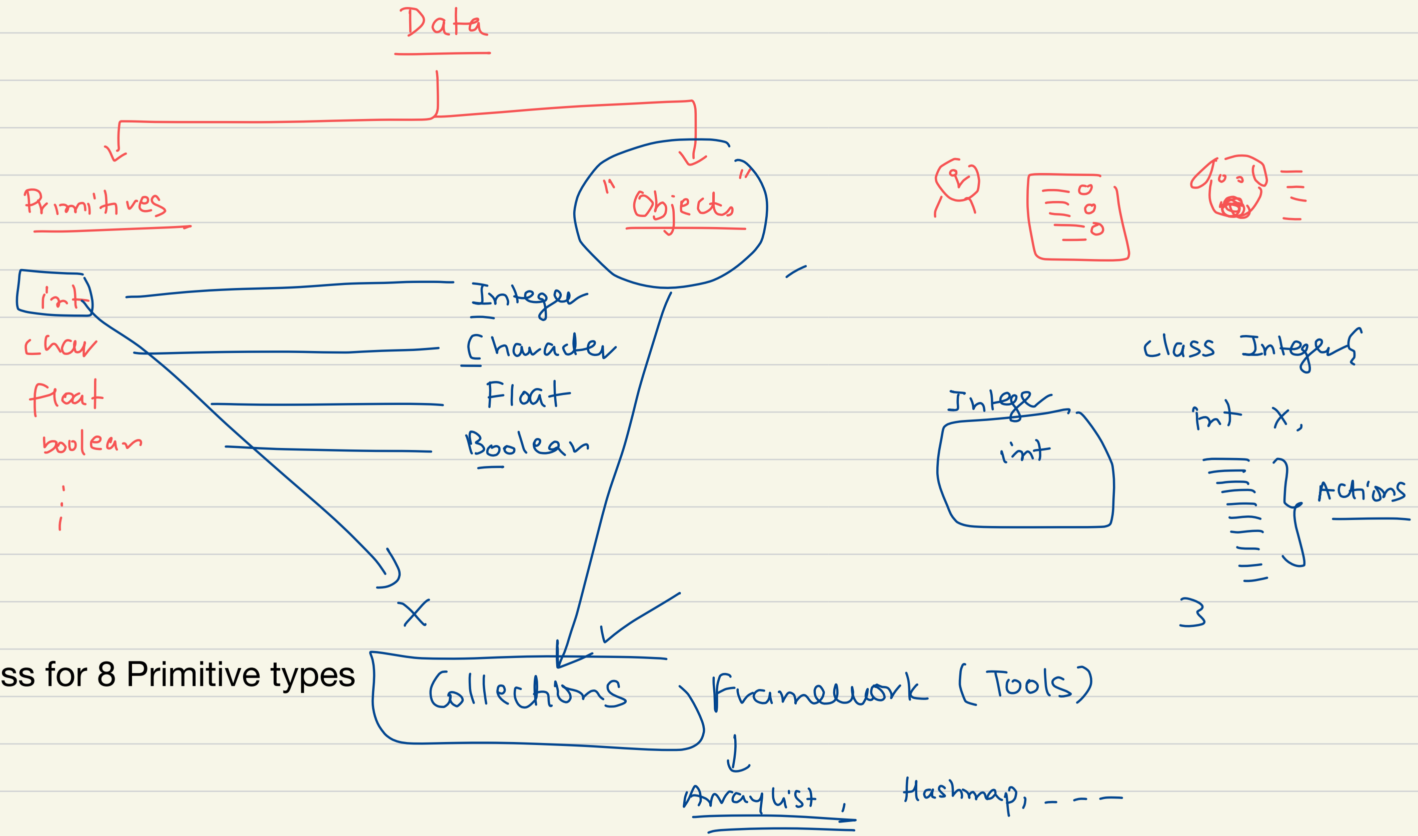
Alarm

time
start-time
set
} Knows

snooze()
SetAlarm()
} actions

Integer
Character
Boolean
Float
Long
Short
Byte
Double

8 Wrapper Class for 8 Primitive types



Data
Primitives
int
char
float
boolean
"Objects"
Integer
Character
Float
Boolean
X
Collections
Framework (Tools)
Arraylist,
Hashmap, - - -
Integer
int
class Integer{
int x,
Actions
}