Array & Arraylist

Object that stores a fixed size, sequential collection of elements of the same data type.

Creation

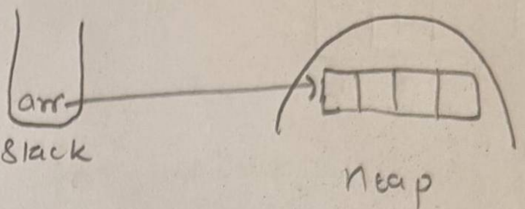
```
int[] arr = new int[5];
int[] arr = {1, 2, 3, 4, 5};
```

Working?

compile time — run time

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

data type — refer. var. — Creation the object in heap.



① Array objects are in heap

② Heap objects are not continuous.

③ DMA

So, in Java may not be continuous
→ Because of JVM.

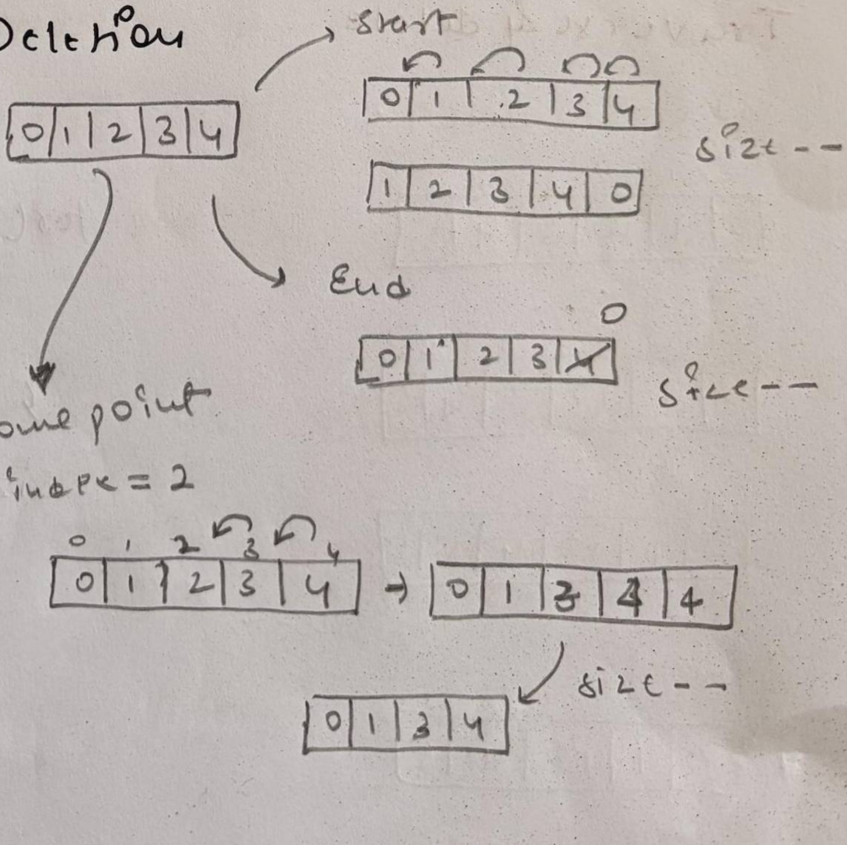
By Default

```
int[] arr = new int[3] → [0, 0, 0]
String[] arr = new String[2] → [null, null]
```

Input

```
arr[0] = 1;
or
for(int i=0; i<arr.length; i++){
    arr[i] = i+1;
}
```

Deletion



Read

```
for(int i=0; i<5; i++){
    sy.out.print(arr[i]);
}
        or
for(int num : arr){
    sy.out(num);
}
        or

sy.out.print(Array.toString(arr));
```

Traverse

```
int size = 0;
int[] arr = new int[10];
arr[0] = 0;
arr[1] = 1;
arr[2] = 3;
arr[3] = 4
arr[4] = 5
size = 5;
```

Traverse & insert

index = 2; → insert 10

0	1	3	4	5
0	1	2	3	4

size++;

0	1	3	4	5	0

shift

0	1	3	4	5	0

0	1	3	3	4	5

0	1	10	3	4	5

```
size++;

~~for (int index = index; index<size-1;~~
~~for (int i = index; i>=0; i--){~~
for(int i = size-1; i>=index; i--){
        arr[i] = arr[i-1];
}
        arr[index] = 10;
```

Traverse & delete

index = 3; delete.

0	1	2	3	4	5
0	1	10	3	4	5

0	1	10	3	4	5

0	1	10	4	5	5

size--

0	1	10	4	5

```
for(int i=index; i<size-1; i++){

        arr[i] = arr[i+1];

}
size--;
```

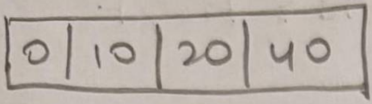
Searching

Time complexity:

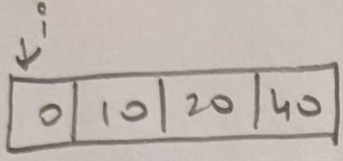
Best ⇒ $O(1)$

Worst ⇒ $O(n)$.

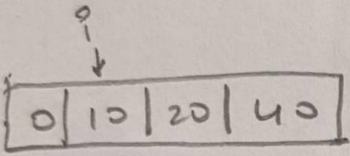
Linear Search



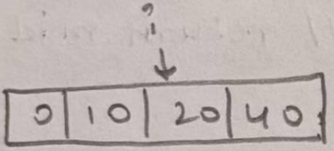
find 20.



arr[i] == 20
// false
i++



arr[i] == 20
// false
i++



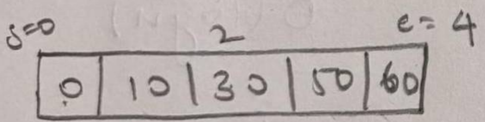
arr[i] == 20
// true
return i;

```
for(int i=0; i<arr.length; i++){
        if(arr[i] == key){
                return i;
        }
}

return -1;
```

When sorted ⇒ Binary Search

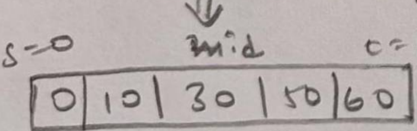
find = 50



⇓

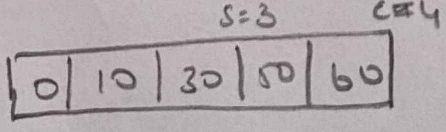
mid = start + (end - start) / 2

$0 + \frac{4-0}{2} = 2$

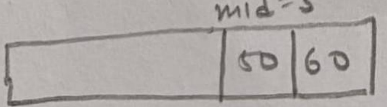


⇓

s = mid + 1   // as mid < target



mid = $3 + \frac{(4-3)}{2}$
= 3

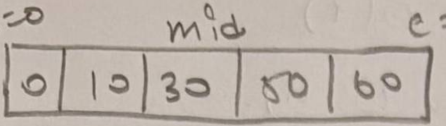


mid == key // true then return.

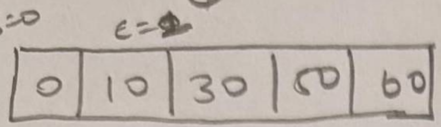
Again ?

find = 10;

// mid we know ⇒ 2

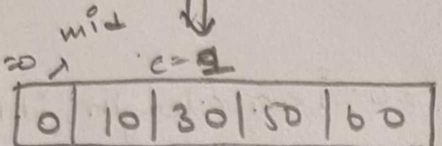


⇓



e = mid - 1 // as mid > target

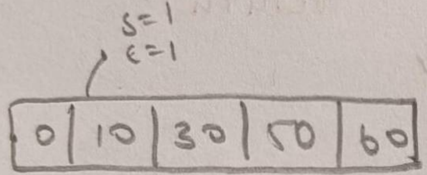
⇓



$$mid = s + (e-s)/2$$
$$= 0 + \frac{(1-0)}{2}$$
$$= 0$$

mid < target

s = mid + 1



$$mid = 1 + \frac{(1-1)}{2}$$
$$= 1$$

mid == target // return mid

Code

```
int start = 0;
int end = arr.length-1;
while( start <= end) {
    // mid
    int mid = start + (~~start~~ End - start)/2

    // mid > target
    if(arr[mid] > target){
        start = mid-1;
    }
    // mid < target
    else if (arr[mid] < target){
        end = mid+1;
    }
    // mid == target
    else {
        return mid;
    }
```

Time complexity
O(logn)

Arraylist

* Non coutinuous memory allocation
* Variable size
* Only objects can be stored

Create

```
Arraylist<Integer> list = new Arraylist<>();
```

Add

```
list.add(3);
list.add(4);
```

At particular index

```
list.add(0, 10);
```

Get

```
list.get(0)     // 10
```

Delete

```
list.remove(0)
```

Sorting

```
Collections.sort(arr);
```