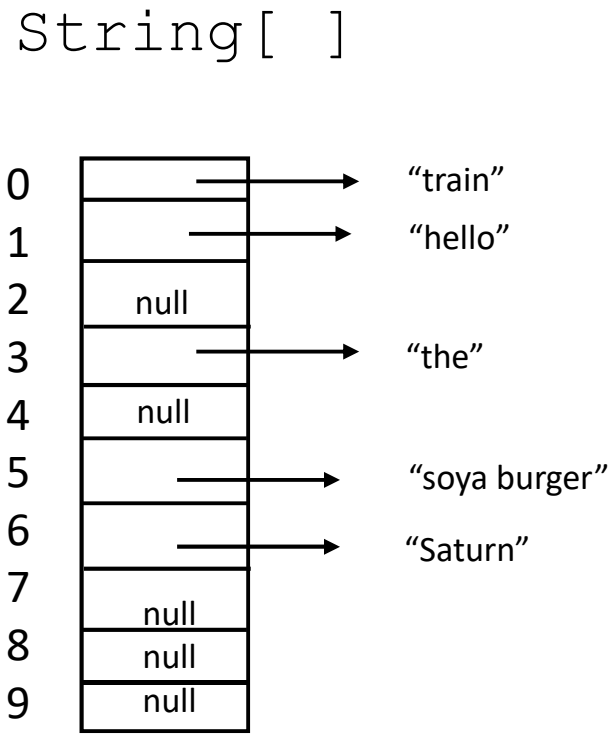
COMP 250

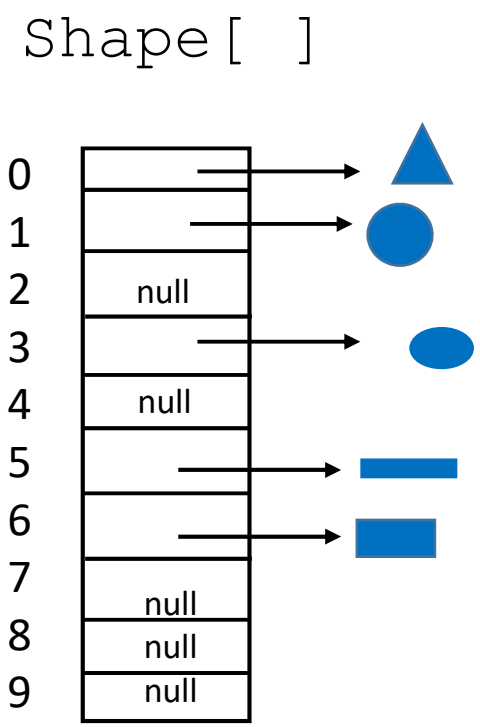
Lecture 9

Array lists

Jan. 26, 2022

Arrays of reference type variables






```
Shape[] shapes = new Shape[428];
        shapes[3] =  new Shape(  );
//  etc...
```

Arrays versus ‘Array Lists’

Arrays contain elements. There is no restriction on *where* the elements are stored.

Elements can be accessed is “constant time”.

Arrays *can be used* to make lists, sometimes called ‘array lists’.

Java has an `ArrayList` class.

List

A list is an ordered set of elements

$$a_0 , a_1 , a_2 , a_3 , \ldots , \ a_{N-1}$$

N is the number of elements in the list, usually called the "size" of the list.

What things do we do with a list?

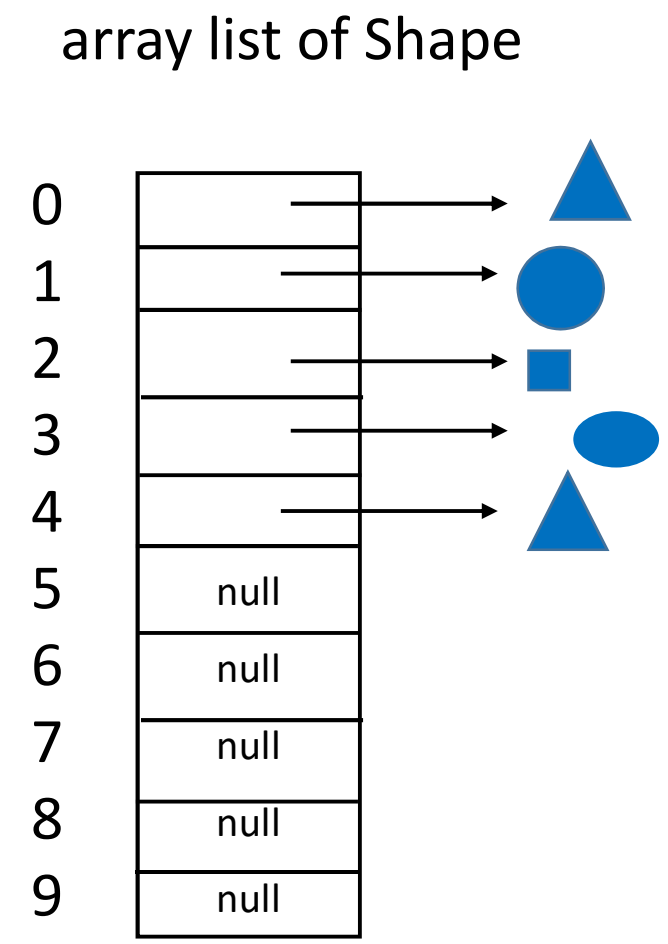
```
get(i)          //  Returns the i-th element (but doesn't remove it)
set(i,e)        //  Replaces the i-th element with e
add(i,e)        //  Inserts element e into the i-th position
remove(i)       //  Removes the i-th element from list
remove(e)       //  Removes first (if any) occurrence of element e
clear()         //  Empties the list.
isEmpty()       //  Returns true if empty, false if not empty.
size()          //  Returns number of elements in the list
……
```

Lists

- arraylist (today)
- singly linked list
- doubly linked list
  :

next two lectures

size = 5
length = 10



How to implement various operations ?
(pseudocode only -- no return type given)

get(i)

set(i,e)

add(i,e)

remove(i)

remove(e)

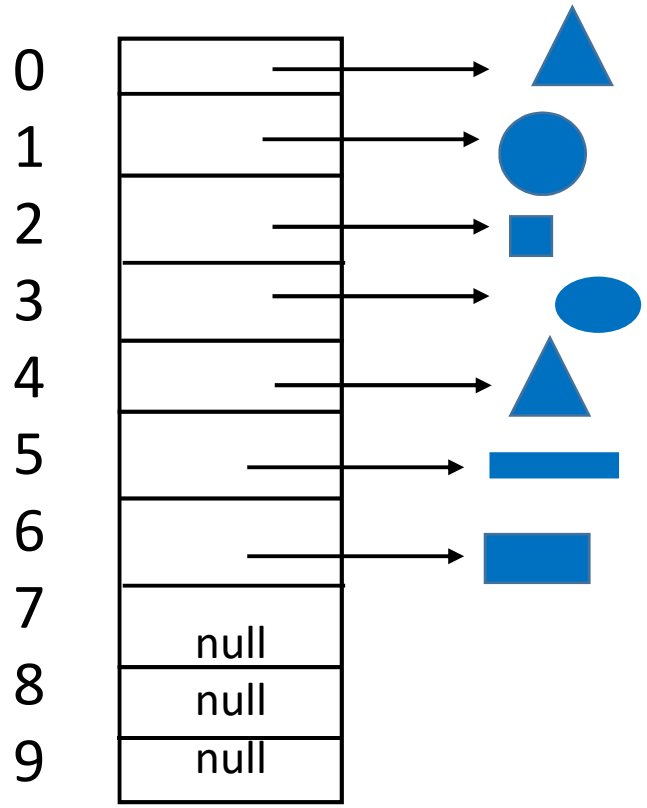
clear()

isEmpty()

size()

…..

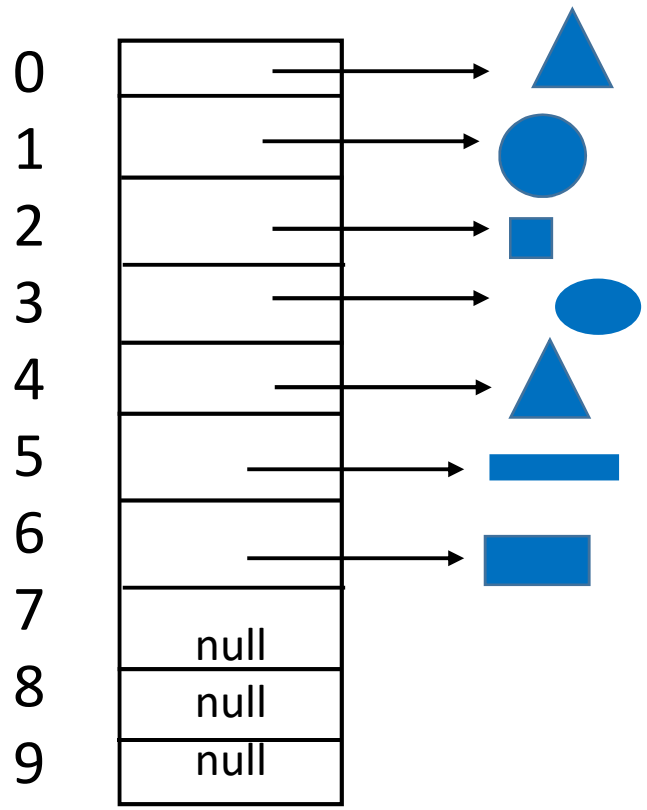
```
get(i) {
        ?
}

//    Let a[ ]  be the array.
```



size = 7

length = 10

```
get(i) {
    if (i >= 0) &  (i < size)
        return a[ i ]
}
```

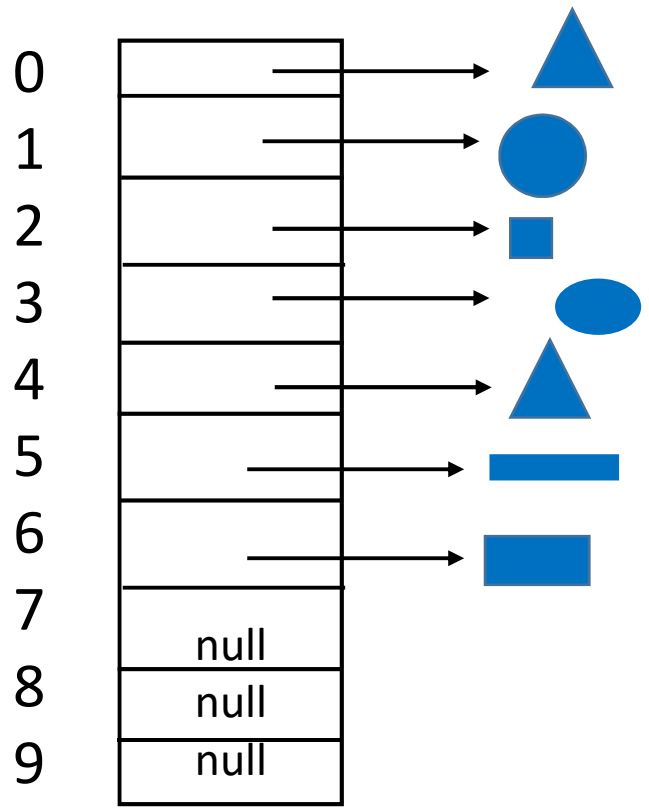


size = 7

length = 10

```
get(i) {
    if (i >= 0) &  (i < size)
        return a[ i ]
    else
        //  throw an index
        //  out of bounds
        //  exception
}
```

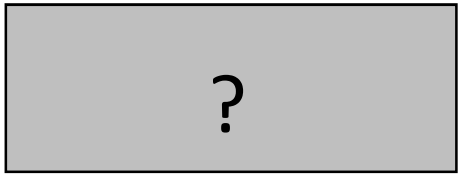
I will not mention this check in the rest of the methods today. But be aware that it would be added in proper implementation.



size = 7

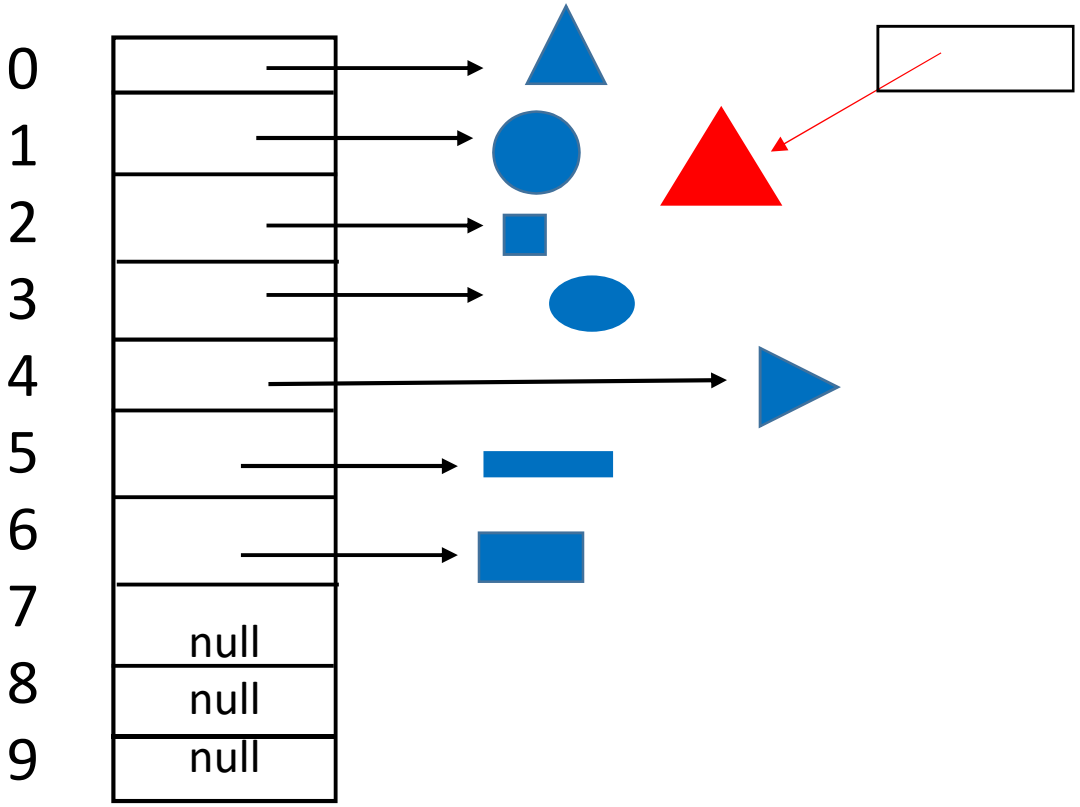
length = 10

set(i,e){ // replaces the object at index i with object e



}

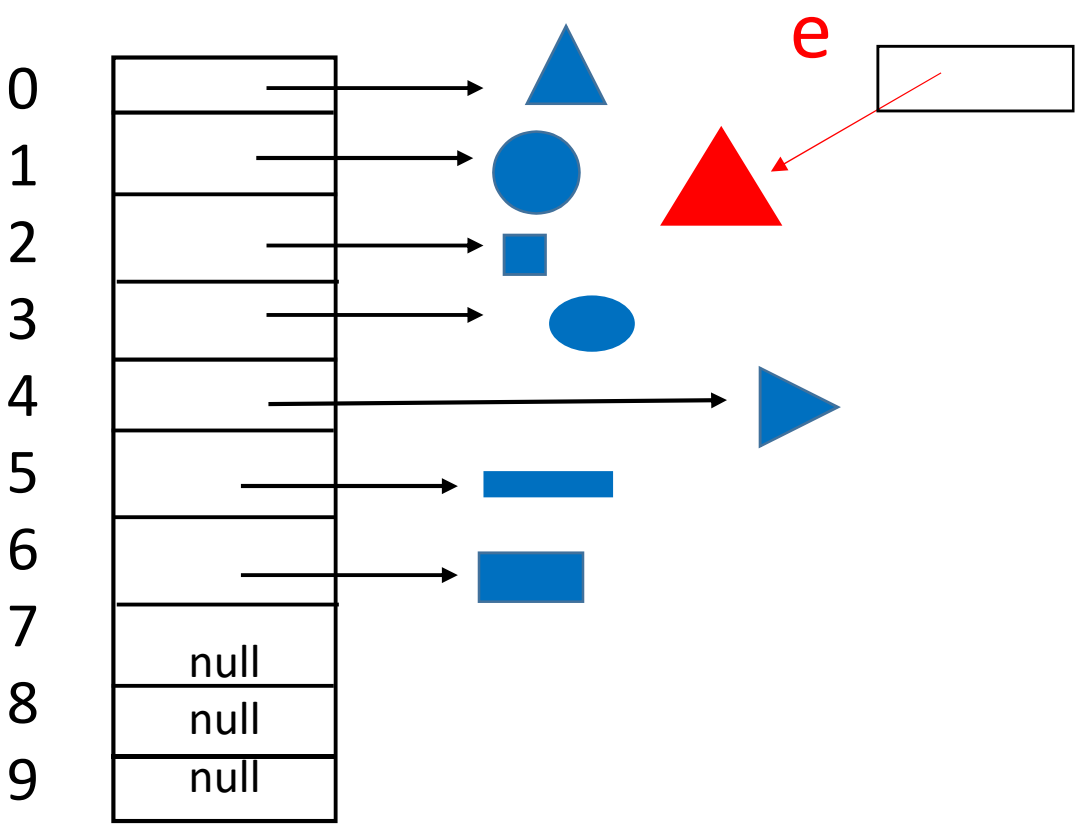
e.g. set(4, e)



I won't draw this box in the rest of the slides

```
set(i,e){    // replaces the object at index i with object e

    if (i >= 0) &  (i < size)
        a[i] = e
}
```

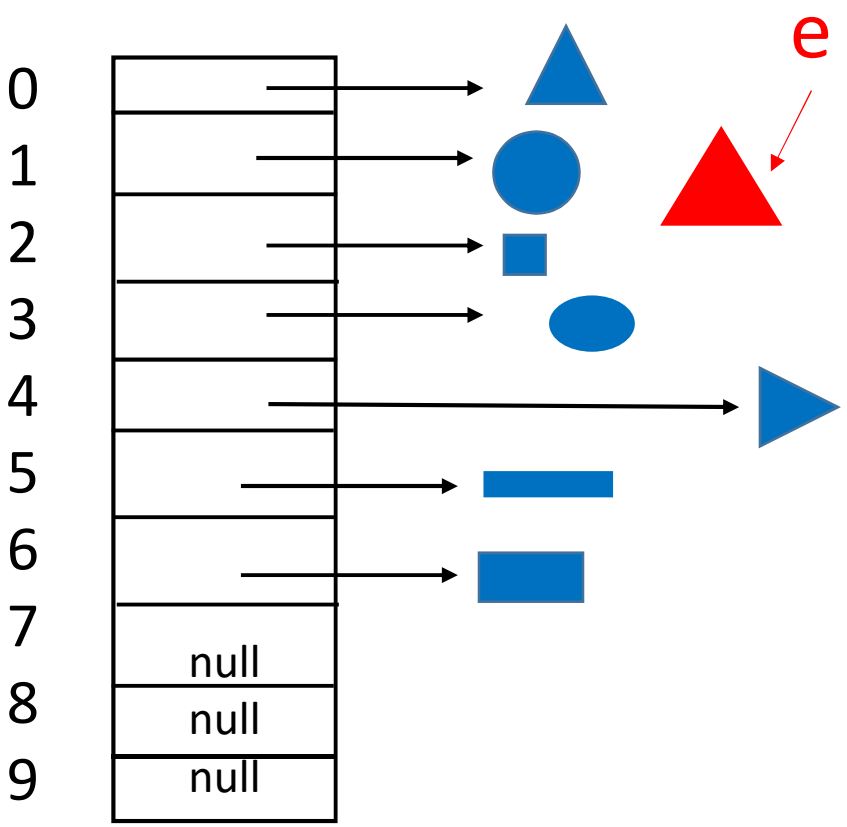
e.g. set(4, e)



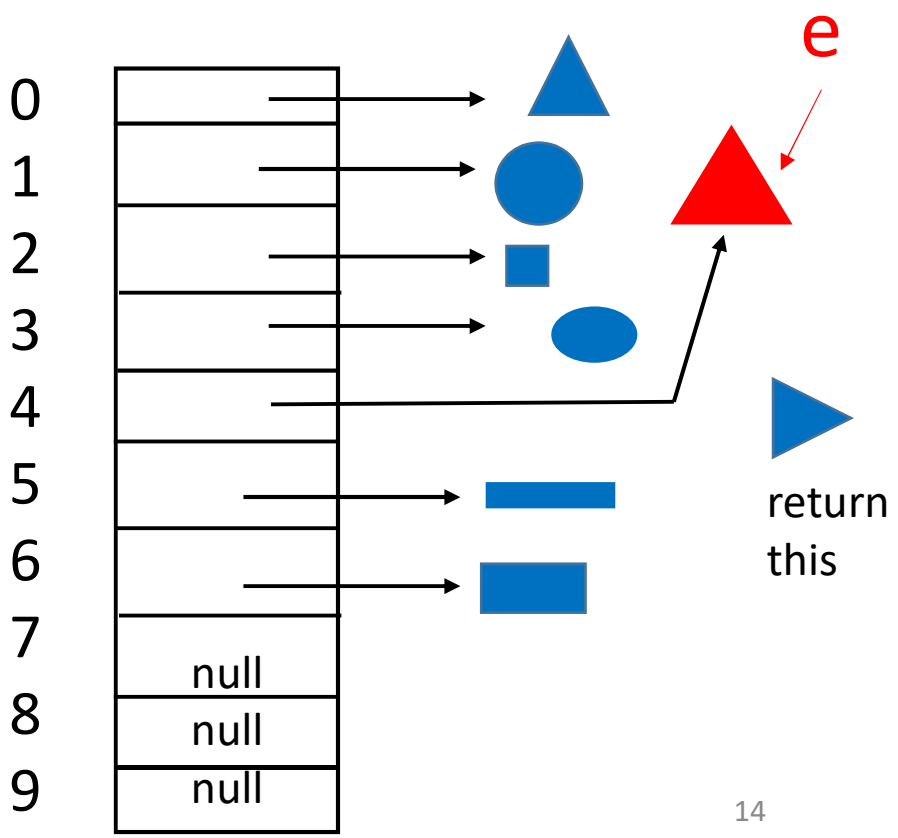
I won't draw this box in the rest of the slides

set(4 ,e)

BEFORE



AFTER



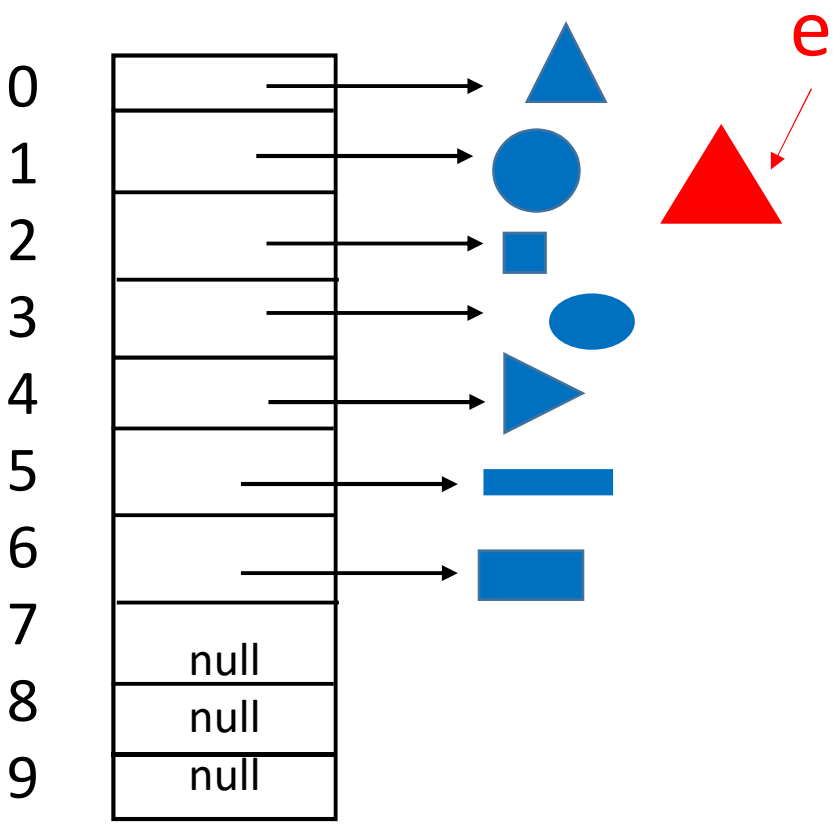
Alternatively, …

```
set(i,e){       //  replace the object at index i
                //   and return the element that is replaced.


    if (i >= 0) &  (i < size)  {
        tmp = a[i]
        a[i] = e
    }
    return tmp
}
```

add(i, e) // also known as “insert”

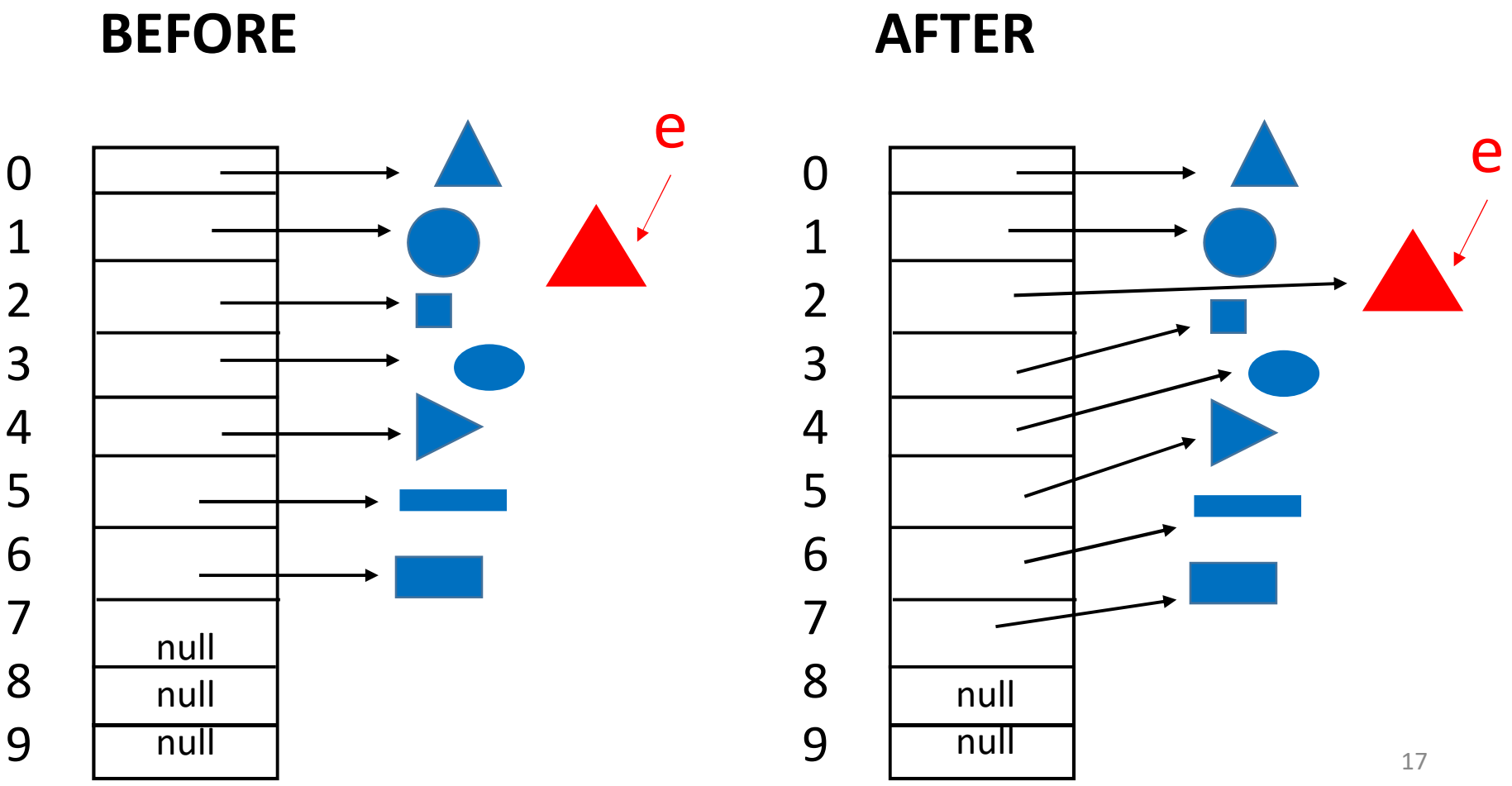
Make room by shifting the references, and then add reference to e

e.g. add(2, e)



add(2, e) // also known as “insert”

Make room by shifting the references, and then add reference to e



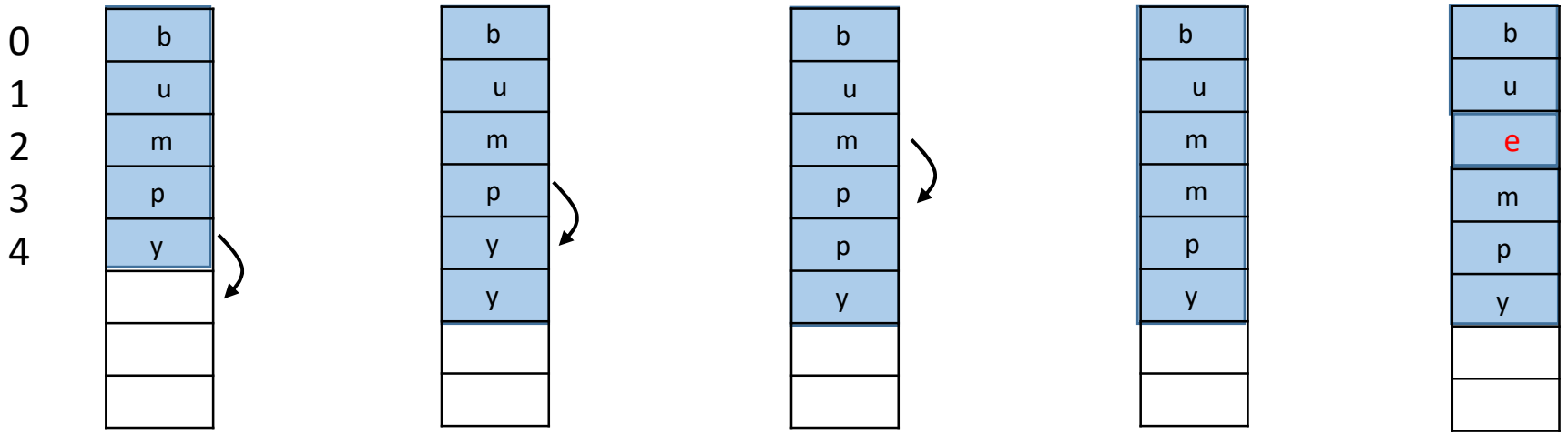
Heads up. I am using e both for parameter name and value here.

add(i, e) e.g. add(2, e)

Similar example, but now I store characters instead of shapes.

Insert character e into slot 2 of array list of characters: bumpy.

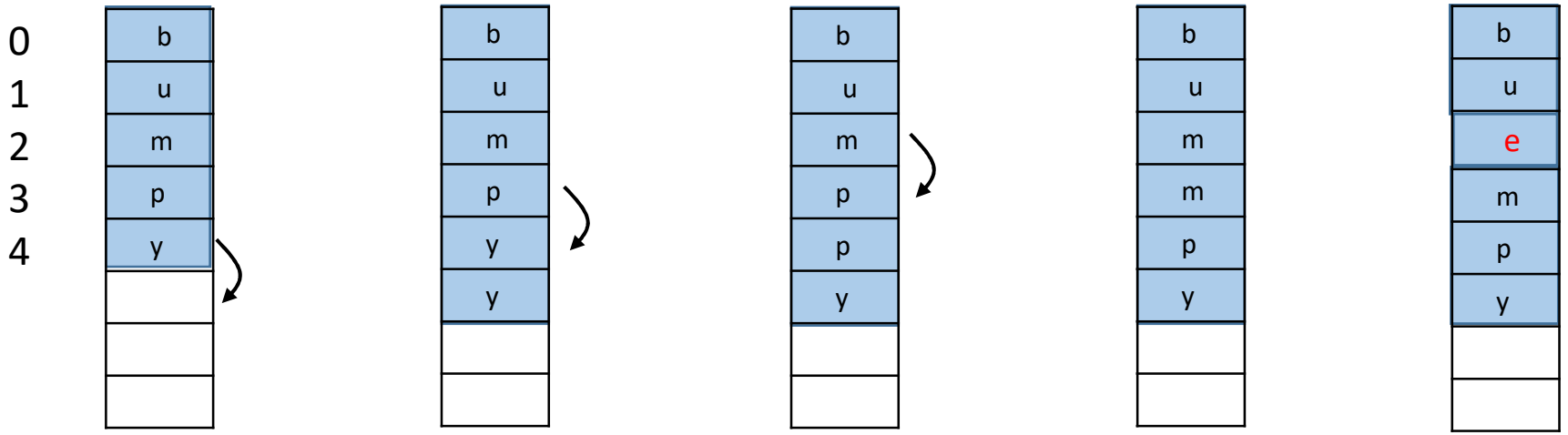
How? Make room by shifting. Then copy in the letter e .
This yields the list: buempy.



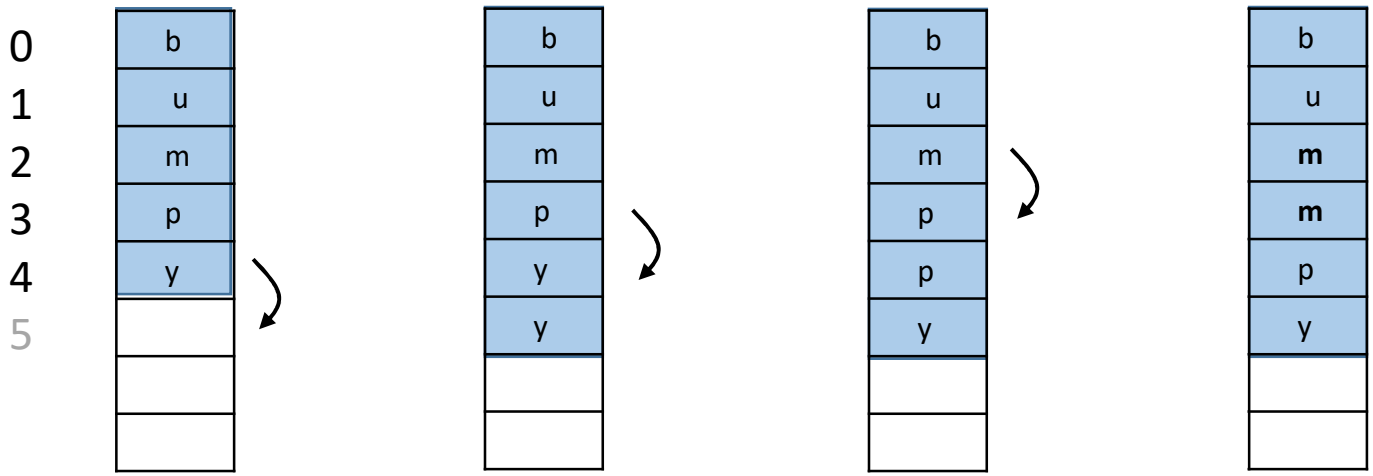
```
add( i, e) {                          //  in the figure below,  i = 2,  size = 5

    if   (i >=0) & (i <= size)   {
                    //  todo:  deal with case that  size == length
```

recall forward shifting pseudocode from lecture 4

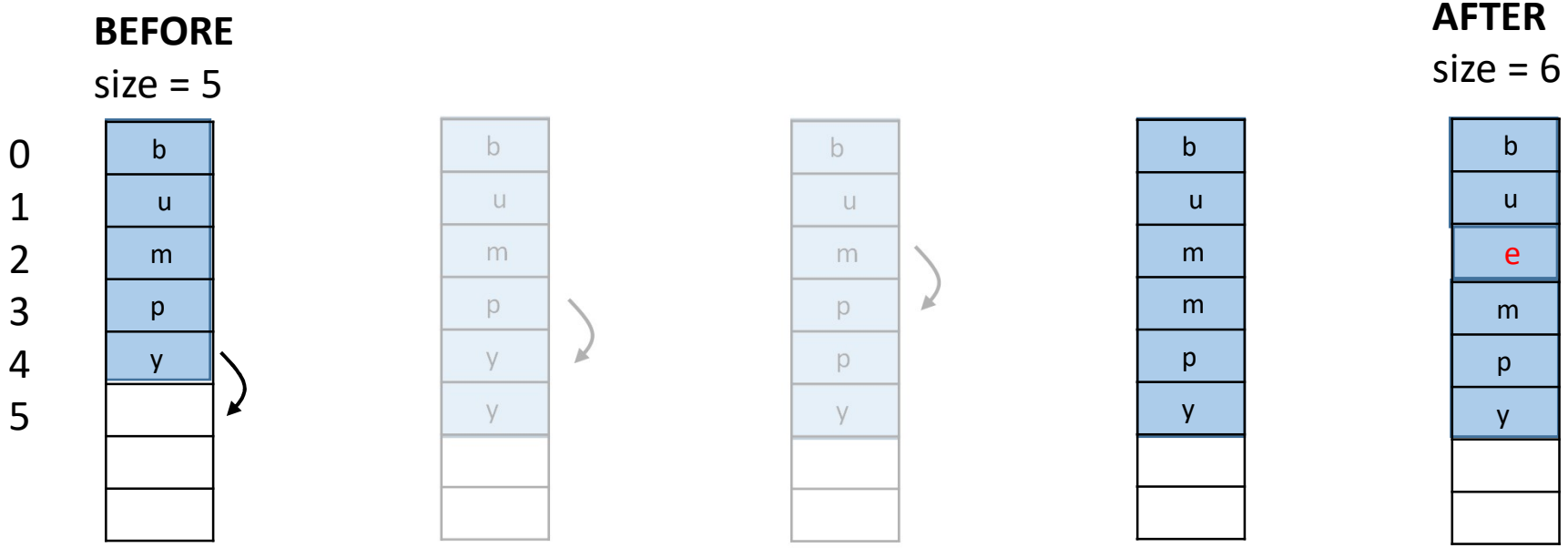
```
    }
}
```



```
add( i, e) {                          //  in the figure below,  i = 2,  size = 5

    if   (i >=0) & (i <= size)  {
         for (k = size; k > i;  k--)
              a[k] = a[k-1]                //   sequence of copy forwards

    }
}
```



```
add( i, e) {                          //  in the figure below,  i = 2

    if   (i >=0) & (i <= size)  {
        for (k = size; k > i;  k--)
            a[k] = a[k-1]

        a[i] = e
        size = size + 1
    }
}
```



How to add an element to an array list when array is full (size == length) ?

```
add( i, e) {

        if (a.size == a.length){              //  is array full?
            make new bigger array b           // e.g.   b.length = 1.5*a.length
            for ( i=0;  i < size;  i++)
                  b[i] = a[i]                 // copy elements to b

           a = b
        }

      //  insert here the add( i , e ) code from last slide
}
```

Suppose you want to add an element to the list and you don't care where it goes?

Use add(e), which calls add(size, e). This adds to the end of the list.

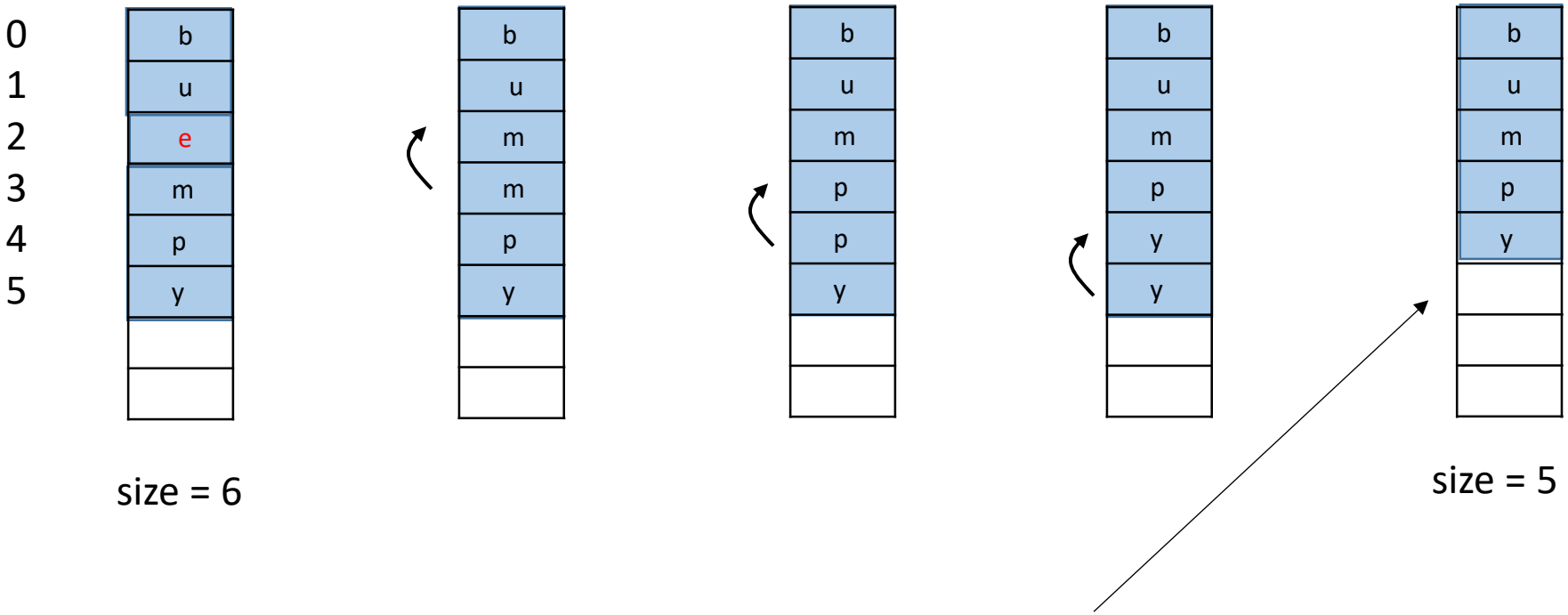
If size < length, then this requires no shifting. It is therefore done in "constant time" (independent of number of elements in the list).

List Operations (`ArrayList`)

```
get(i)
set(i,e)
add(i,e)
remove(i)      //  Removes the i-th element from list
remove(e)      //  Removes element e from the list (if it is there)
clear()        //  Empties the list.
isEmpty()      //  Returns true if empty, false if not empty.
size()         //  Returns number of elements in the list
  :
```

remove(i) // e.g. i = 2 (removes the letter ‘e’)

It does the opposite of the add(i, e) method.



The value ‘y’ will still be there, but it is not accessible. Why not?

remove(i)

```
if ( (i >= 0)  and  (i < size) ){

    tmp = a[i]                          //    put aside and later  return it

    Shift backwards the elements from position i+1 to end

    size = size – 1

    return tmp
}
```

remove(i)

```
if ( (i >= 0)  and  (i < size) ){

     tmp = a[i]                          //    put aside and later  return it

    for ( k = i;  k < size - 1;  k++){
              a[ k ] = a[ k + 1 ]
    }

    size = size – 1

    return tmp
}
```

Method Overloading

```
add( i ,e)      //  inserts element e into the i-th position
add( e )        //  inserts element e at end of list


remove(i)       //  Removes the i-th element from list
remove(e)       //  Removes first occurrence of element e
                //   from the list (if it is there)
```

Java `ArrayList` class

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

It uses an array as the underlying data structure.

When the array is full and a new element is added, it "grows" the array by 50% namely, it creates a new array that is bigger and copies the elements to this new array.

Java generic types

An array of what? ArrayList< T >

Example:

```
ArrayList< Shape >     shape  =  new ArrayList<  Shape >();

        //  initializes the array length (capacity) to 10

ArrayList<  Shape >    shape  =  new ArrayList<  Shape >( 23 );

        //  initializes the underlying array length to 23
```

Java generic types

An array of what? ArrayList< T >

It must be a reference type.

Example:

ArrayList< Integer > shape = new ArrayList< Integer >();

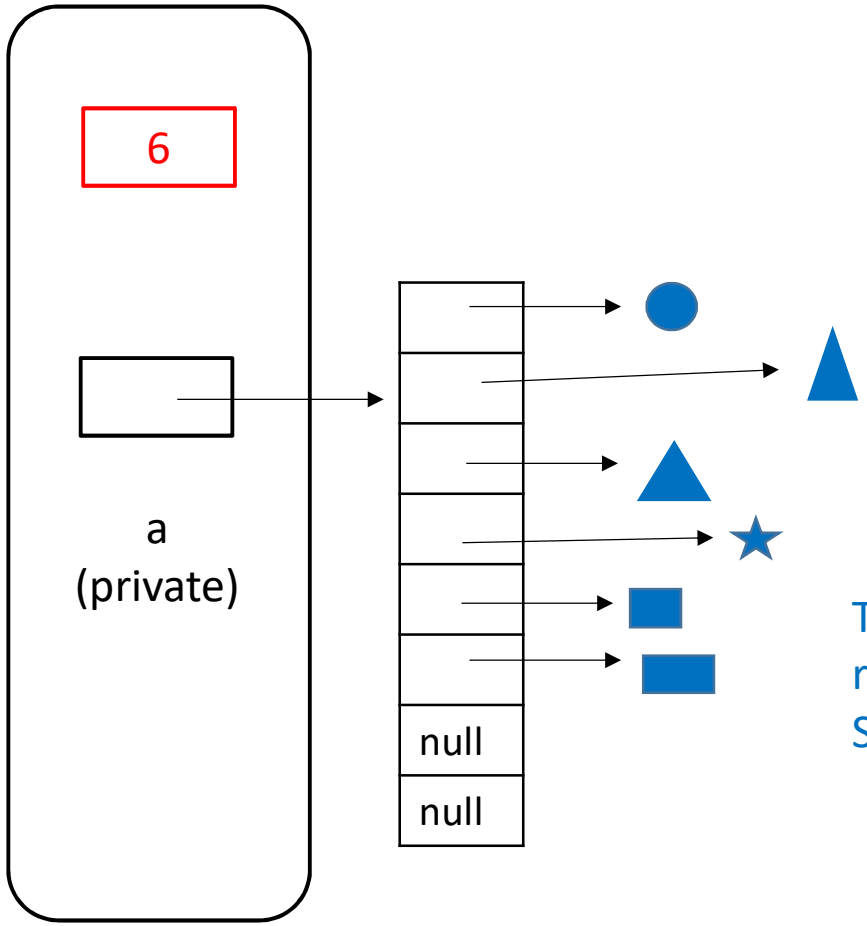
ArrayList< Integer > shape = new ArrayList< Integer >(23);

You cannot use a primitive type (int) in there.

Java `ArrayList` object

Has private field that holds the number of elements in the list (size).

Has a private field that references an array object.



The array object references various Shape objects.

How many objects are there in this figure? 8

Exercise (challenging)

Suppose you want to add N elements to an array list, which is initially empty.

Each time you fill the array, double its length.

(Java's `ArrayList` multiplies length by 1.5, not 2).

What is the O( ) time complexity ?

Specifically…

Q: How many times k do we need to double the length of the array so that it can hold N elements ?

Q: How many copy operations are required ?
(When we increase the array length, we copy from the small array to large.)

See Exercises PDF for today. Try to answer the questions without peeking at the solutions.

Coming up…

Lectures

Fri. Jan. 28

Singly Linked Lists

Assessments

Fri. Jan. 28

Quiz 1 (lectures 1-7)

- practice quiz posted

Assignment 1 to be posted

- you will have 2 weeks to do it