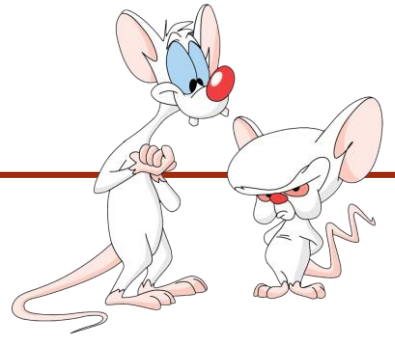
COMP 250

INTRODUCTION TO COMPUTER SCIENCE

Week 5 - Lab

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Arraylist

ARRAYS IN JAVA

- **Arrays whose elements have a primitive type**

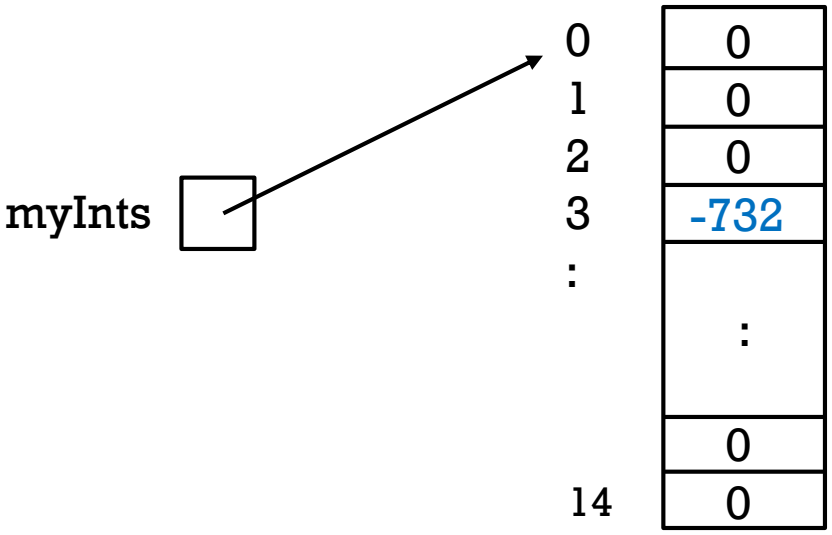
```
int[] myInts = new int[15];
myInts[3] = -732;
```

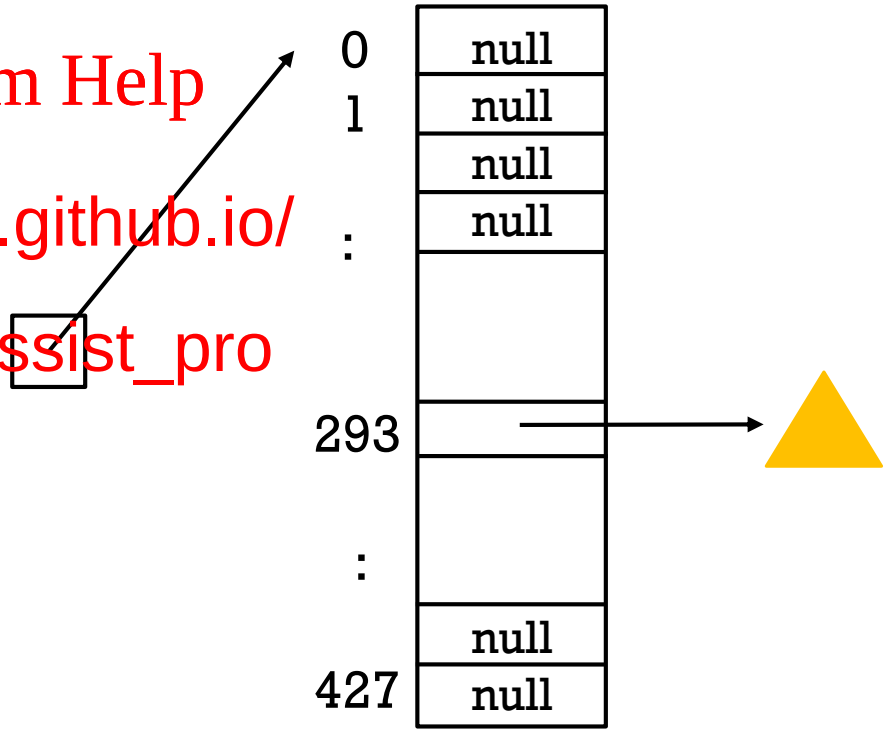

- **Arrays whose elements have a reference**

```
Shape[] myShapes = new Shape[428];
shapes[293] = new Shape(  );
```

```
int[] myInts = new int[15];
myInts[3] = -732;
```

```
Shape[] myShapes = new Shape[428];
shapes[293] = new Shape( );
```



You can think of an array as a block of consecutive slots in memory

ARRAYS HAVE CONSTANT TIME ACCESS

A computer accesses an element in an array in constant time

i.e. constant, inde of the array.


```
....   = a[k];   // re
```



```
a[k]  =  .... ;        // write
```

You will learn more about how this works in COMP 206 and 273.

LIST

An ordered set of elements

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

WHAT WOULD WE LIKE TO 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(e)          // A                                nd of the list
add(i,e)        // In                               -th position
remove(i)       // Removes the                      n list
remove(e)       // Removes first occurrence of 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
```

IMPLEMENTATIONS

There are different implementations of a list:

- Array list
- Singly linked list
- Doubly linked list

IMPLEMENTATIONS

There are different implementations of a list:

- Array list → Use an array to implement a list!
- Singly linked list
- Doubly linked list

ARRAYLIST

Idea:

- Use an array to store the elements of the list
- Keep track of how many ele
  inserted in the list

ArrayList.java

```
public class ArrayList {
    private Shape[] arr;
    private int size;
    ...
}
```

To decide:

- How big should the underlying array be when we first create an object of type ArrayList?
  (this is referred to as the *initial capacity* of the list)

ARRAYLIST

Idea:

- Use an array to store the elements of the list
- Keep track of how many ele
  inserted in the list

To decide:

- How big should the underlying array be when we first create an object of type ArrayList?
  → Java's ArrayList creates an array of length 10.

ArrayList.java

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    public ArrayList() {
        arr = new Shape[10];
        size = 0;
    }
    :
}
```

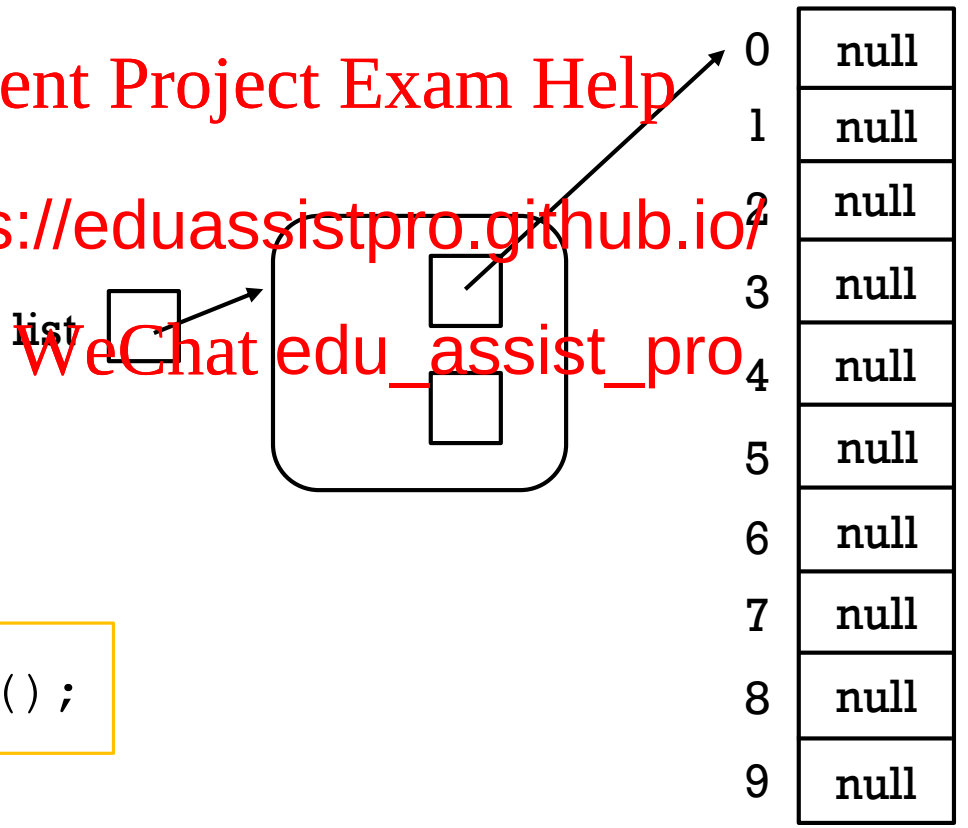
EXAMPLE

ArrayList.java

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    public ArrayList() {
        arr = new Shape[10];
        size = 0;
    }
}
```



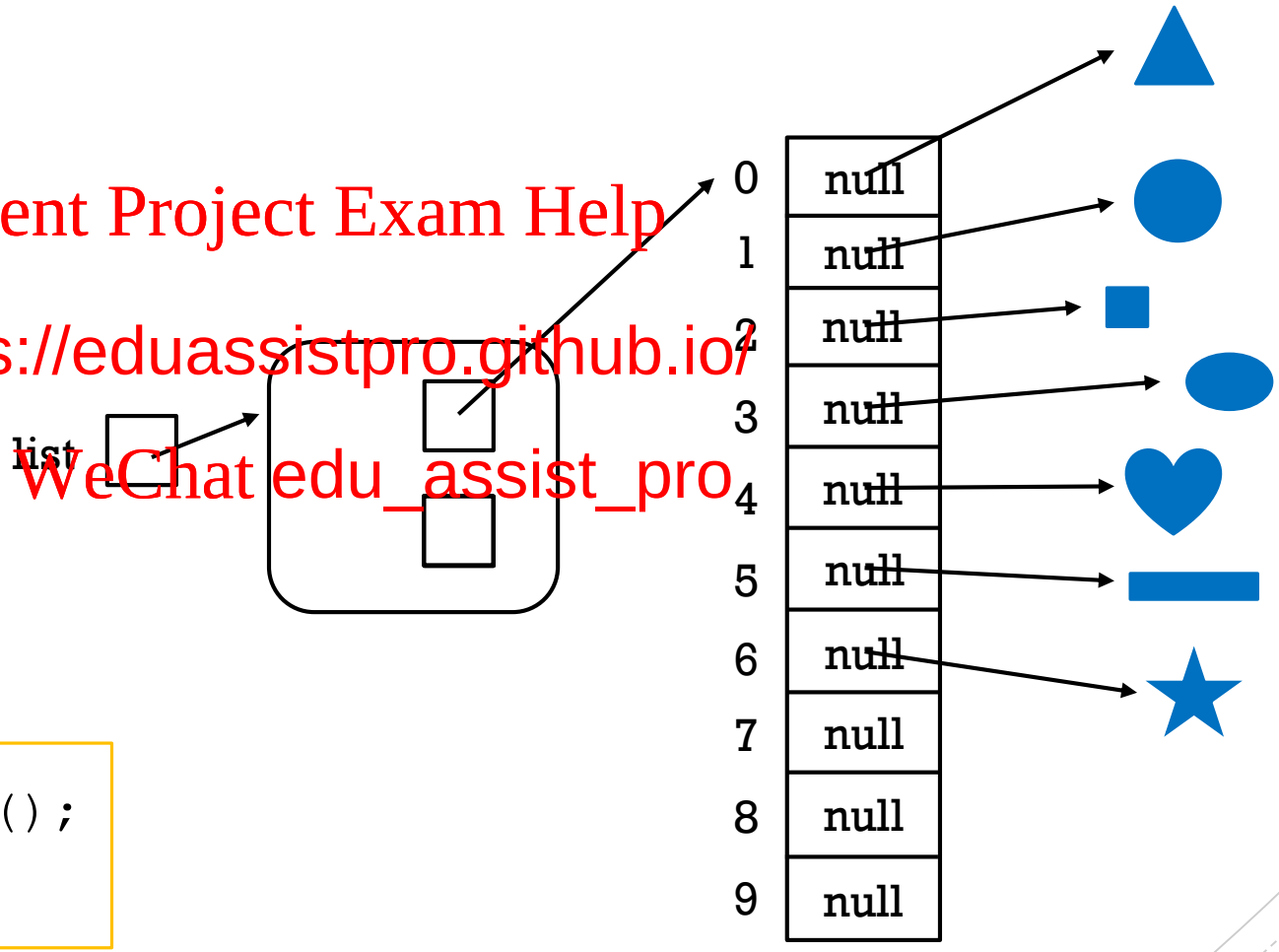
```
ArrayList list = new ArrayList();
```



EXAMPLE – WHAT WE WANT WHEN ADDING ELEMENTS

ArrayList.java

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    public ArrayList() {
        arr = new Shape[10];
        size = 0;
    }
}
```

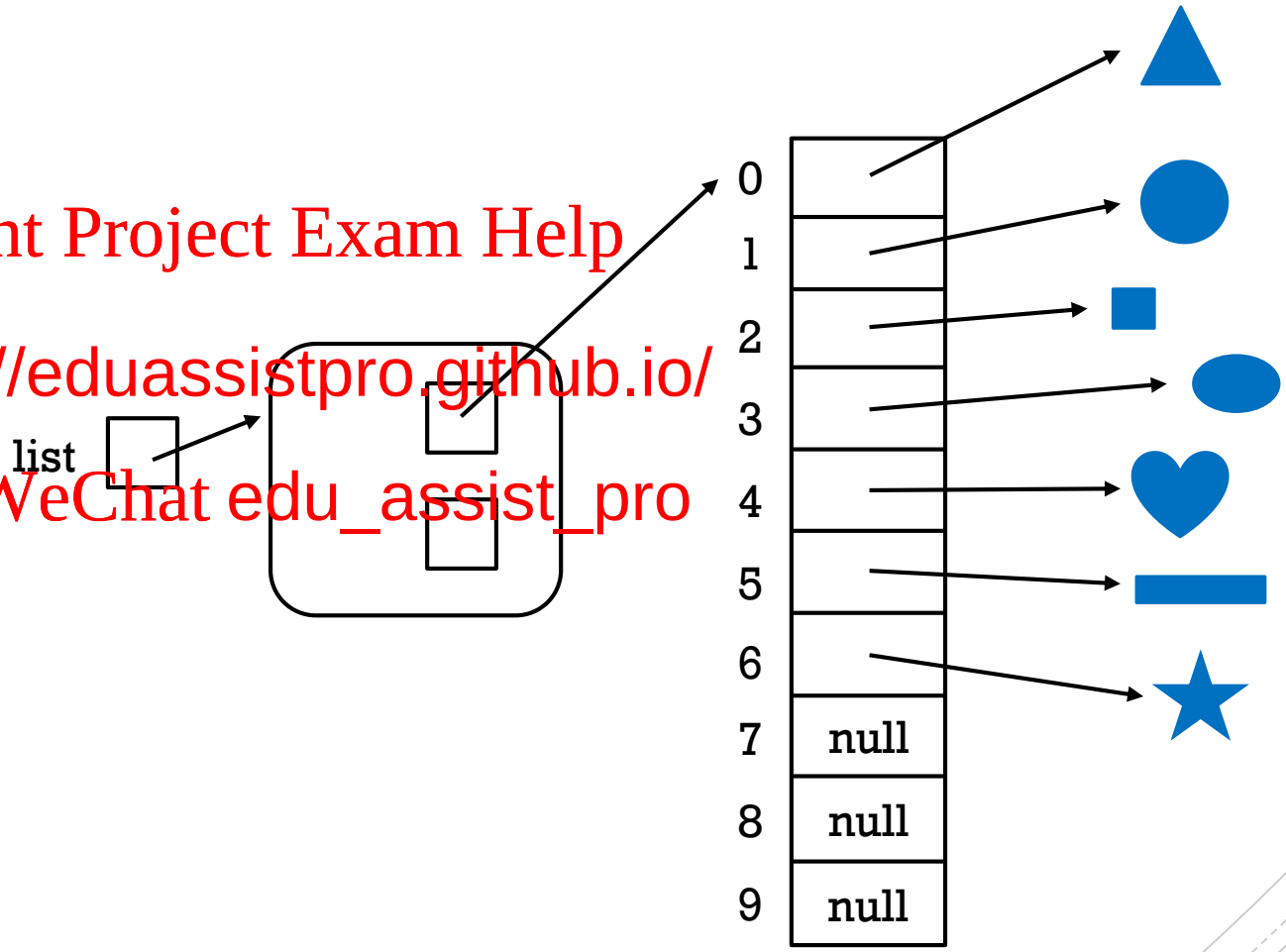




```
ArrayList list = new ArrayList();
// add 7 elements…
```

HOW TO IMPLEMENT VARIOUS OPERATIONS? – `get()`

Returns the element at the specified position in this list.

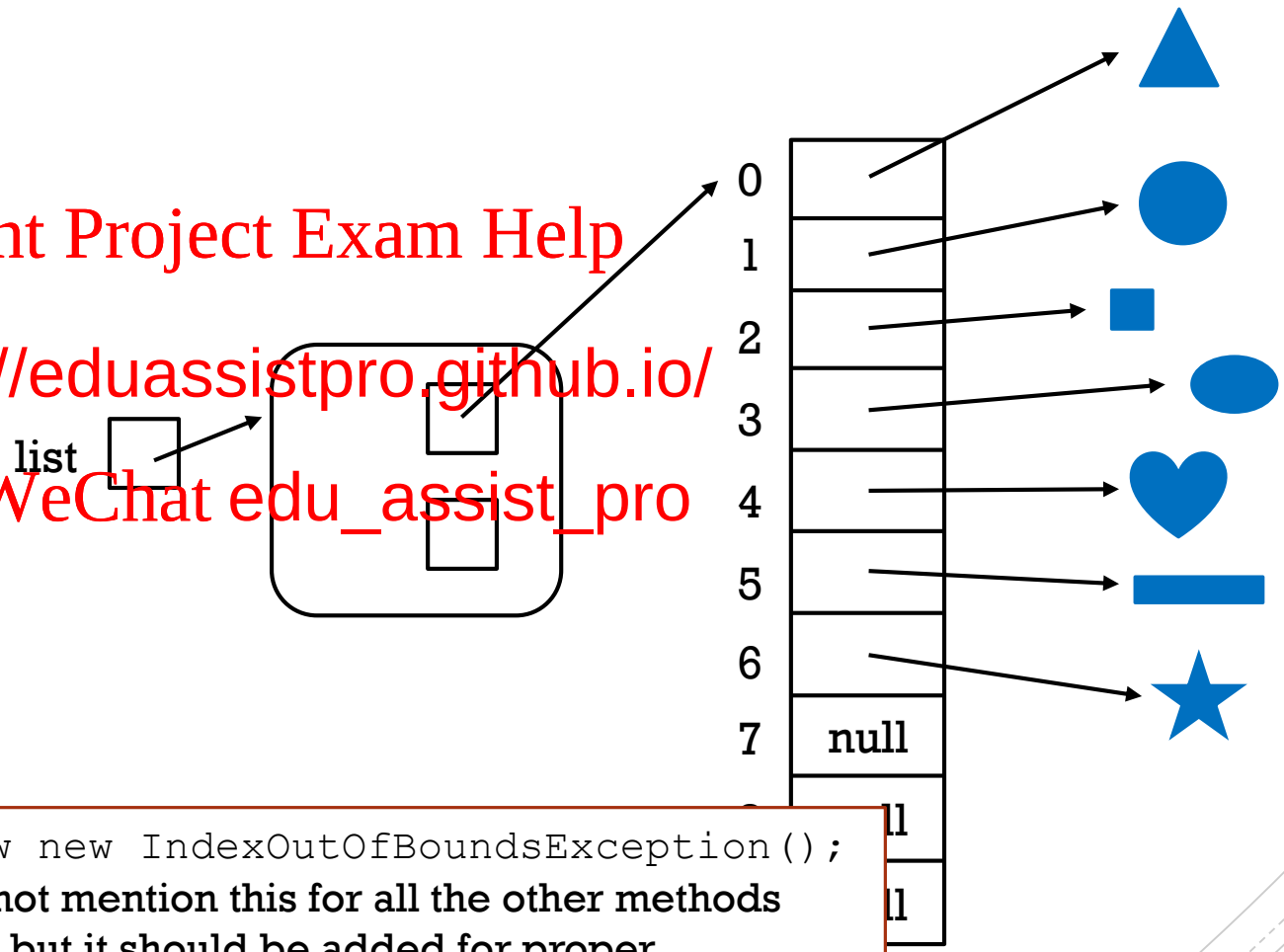
```
public class ArrayList {
    private Shape[] arr;
    private int size;

    :

    public Shape get(int i) {
        if(                    )
            return arr[i];
        // otherwise?
    }
}
```

HOW TO IMPLEMENT VARIOUS OPERATIONS? – `get()`

Returns the element at the specified position in this list.

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    :

    public Shape get(int i) {
        if( i>=0 && i<size )
            return arr[i];
        // otherwise?
    }
}
```

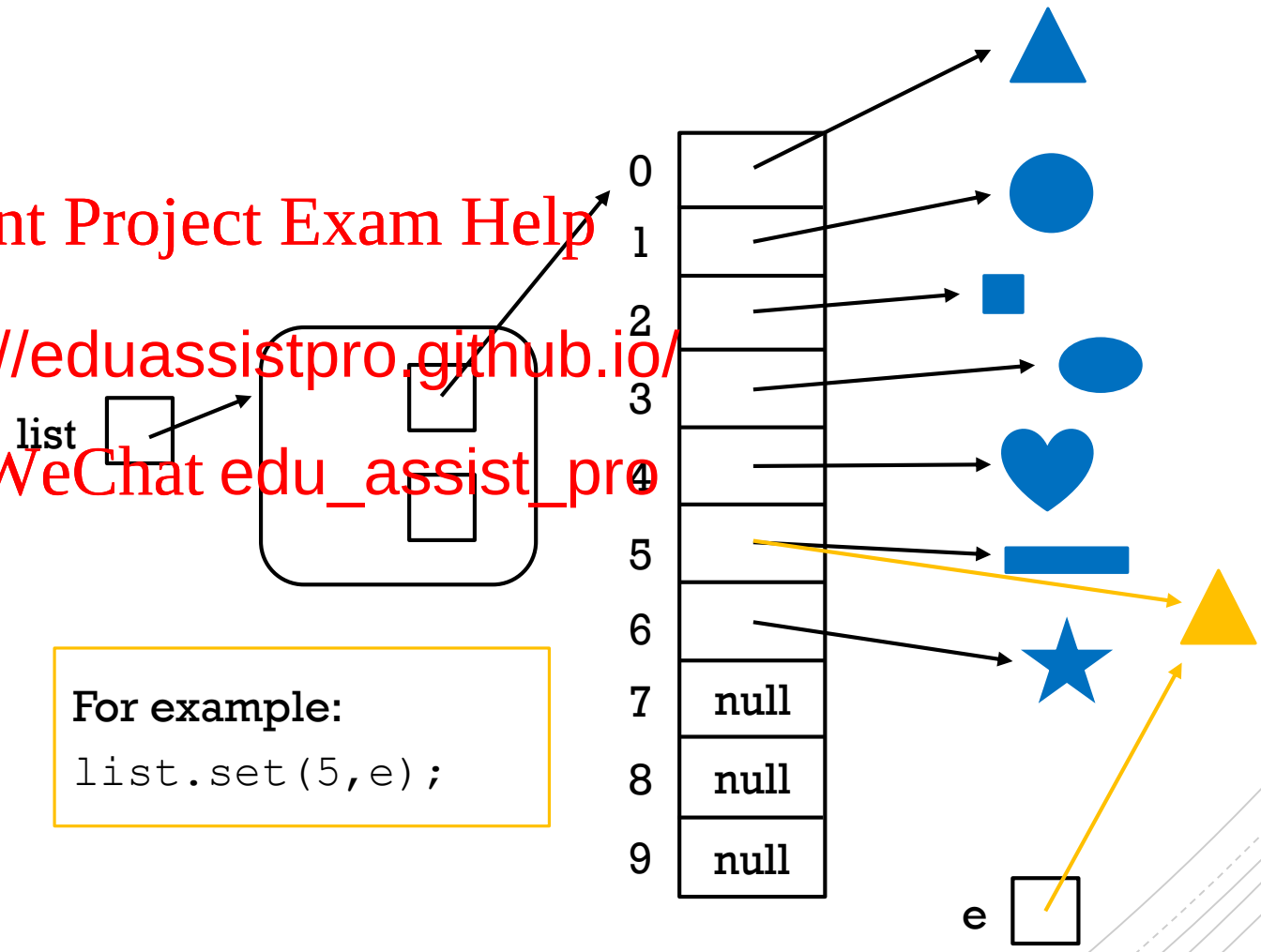

```
throw new IndexOutOfBoundsException();
```

I will not mention this for all the other methods today, but it should be added for proper implementations.

HOW TO IMPLEMENT VARIOUS OPERATIONS? – `set()`

Replaces the element at the specified position in this list with the specified element.

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    :

    public Shape set(int i, Shape e)
    {
        if(i>=0 && i<size) {
            Shape tmp = arr[i];
            arr[i] = e;
            return tmp;
        }
    }
}
```

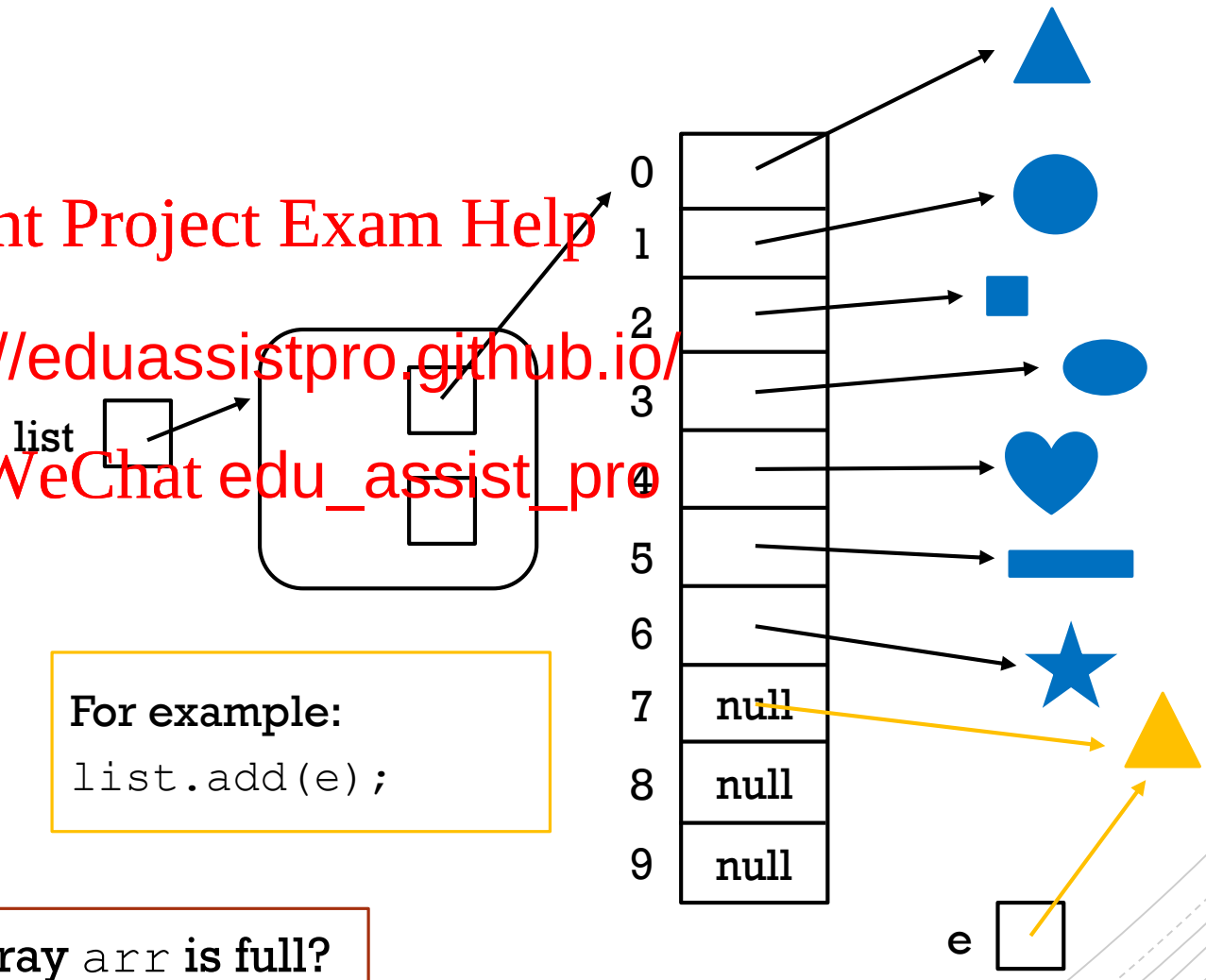
For example:

```
list.set(5,e);
```

HOW TO IMPLEMENT VARIOUS OPERATIONS? – add()

Appends the specified element to the end of this list.

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    :

    public void add(Shape e)
    {
        arr[     ] = e;
        size = size + 1;
    }
}
```

For example:

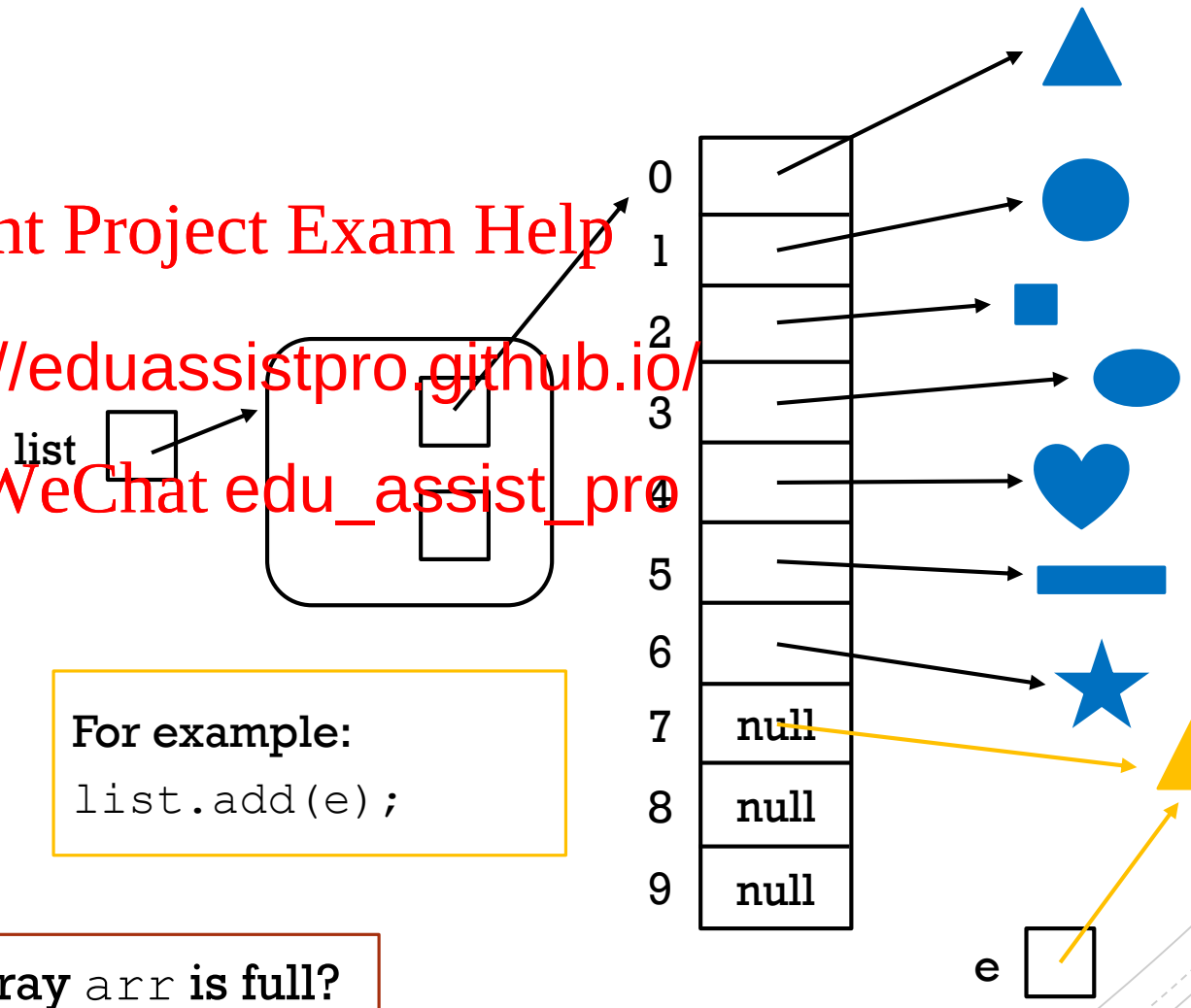
```
list.add(e);
```

What if the array `arr` is full?

HOW TO IMPLEMENT VARIOUS OPERATIONS? – add()

Appends the specified element to the end of this list.

```
public class ArrayList {
    private Shape[] arr;
    private int size;

    :

    public void add(Shape e)
    {
        arr[size] = e;
        size = size + 1;
    }
}
```

For example:

```
list.add(e);
```

What if the array `arr` is full?

HOW TO IMPLEMENT VARIOUS OPERATIONS? – `add()`

Appends the specified element to the end of this list.

What if the array arr is already full?

```
public void add(Shape e) {
    if (          ) {
        resize();
    arr[size] =
    size = size
}

private void resize() {
    Shape[] bigger = new Shape[arr.length*2]; // example
    for(int i=0; i < size; i++) {
        bigger[i] = arr[i];
    }
    arr = bigger;
}
```

HOW TO IMPLEMENT VARIOUS OPERATIONS? – `add()`

Appends the specified element to the end of this list.

What if the array arr is already full?

```
public void add(Shape e) {
    if (arr.length == size)
        resize();
    arr[size] =
    size = size
}

private void resize() {
    Shape[] bigger = new Shape[arr.length*2]; // example
    for(int i=0; i < size; i++) {
        bigger[i] = arr[i];
    }
    arr = bigger;
}
```

OVERLOADING

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



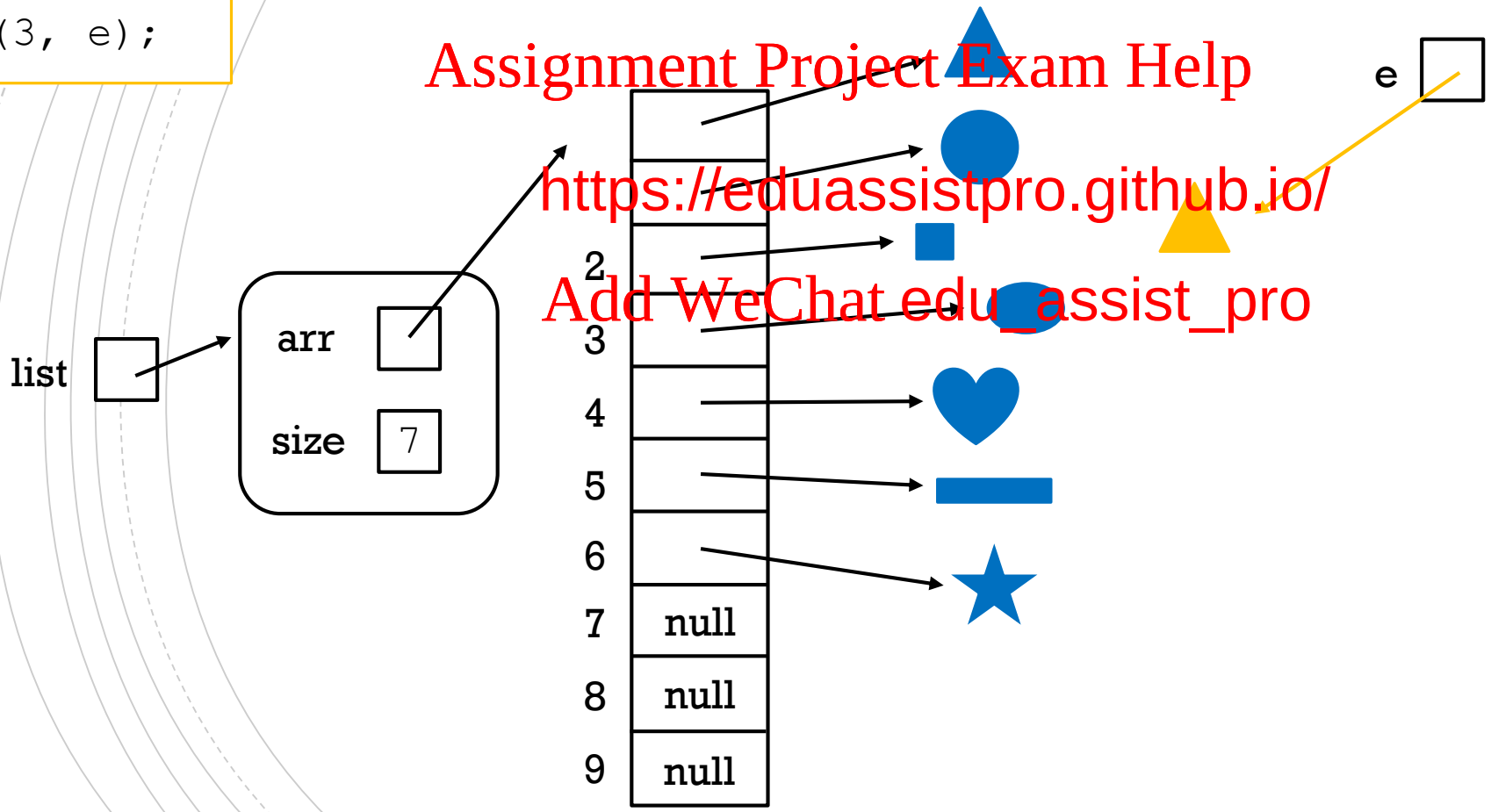
```
remove(i)     //  Removes the i-th element from list
remove(e)     //  Removes first occurrence of element e
              //   from the list (if it is there)
```

HOW TO IMPLEMENT `add(i,e)`

Inserts the specified element at the specified position in the list.

For example:

```
list.add(3, e);
```

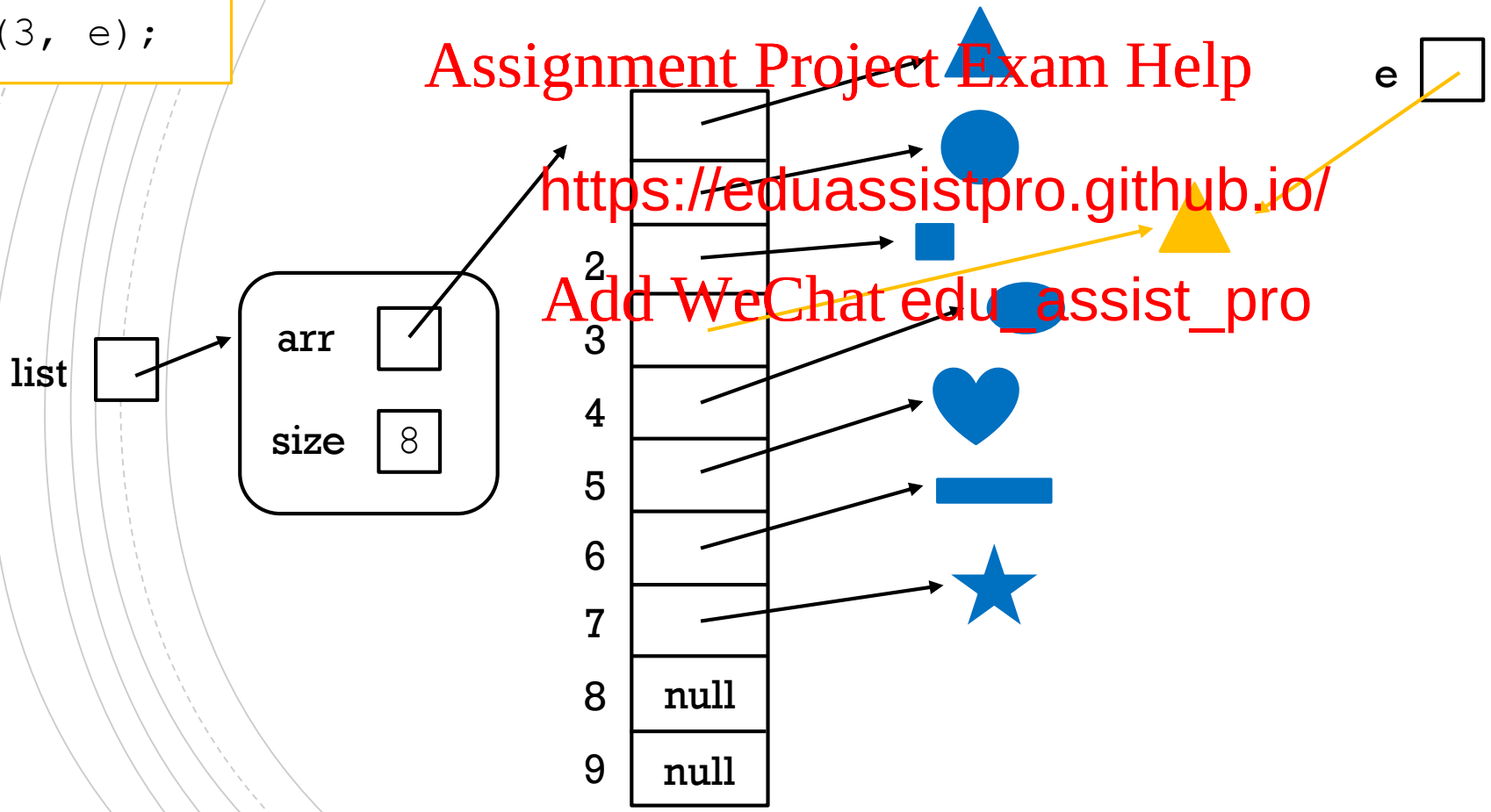
HOW TO IMPLEMENT `add(i,e)`

Inserts the specified element at the specified position in the list.

IDEA: make room by shifting the elements, then add `e`

For example:

```
list.add(3, e);
```

HOW TO IMPLEMENT `add(i,e)`

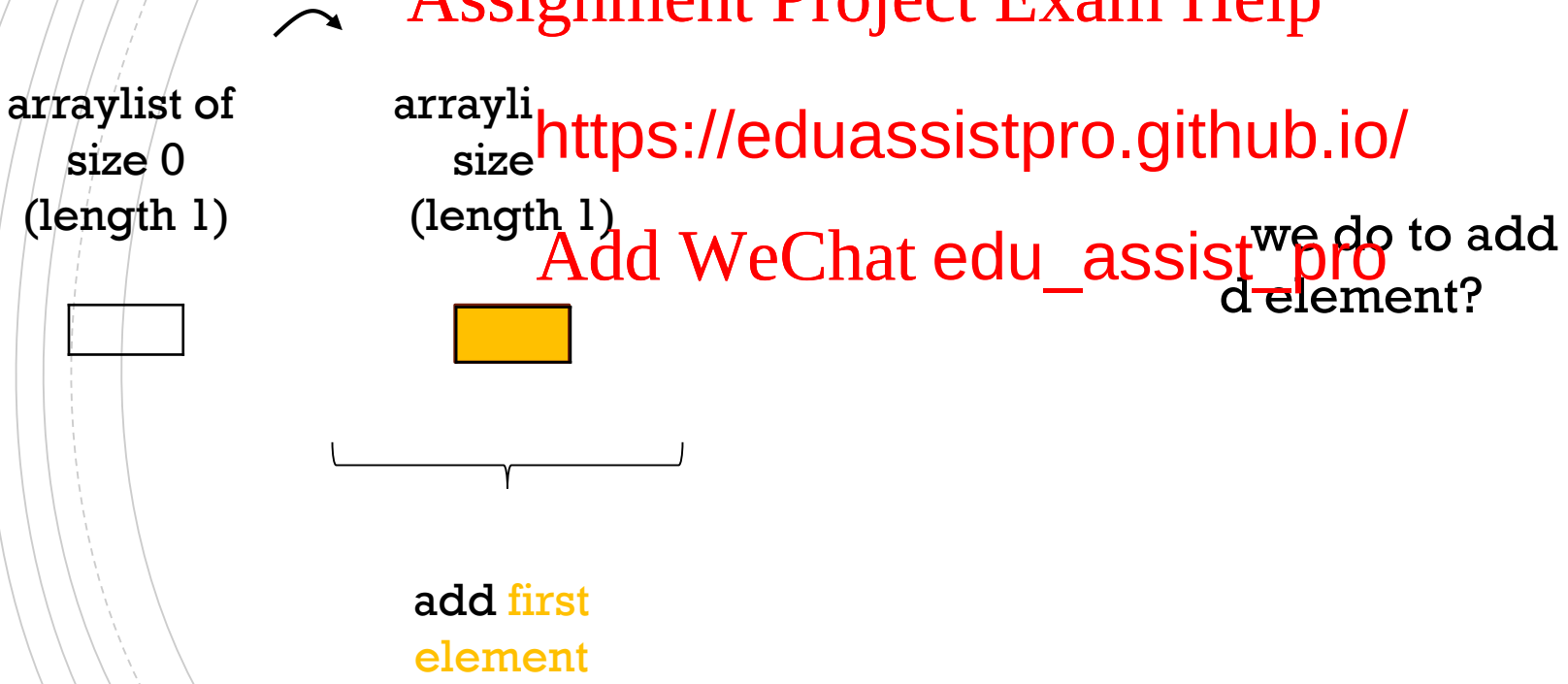
Inserts the specified element at the specified position in the list.

```
public void add(int i, Shape e) {

    // Throw                                of bounds

    // Resize if not enoug

    // Shift elements down


    // Add the new element


}
```

ADDING N ELEMENTS TO AN ARRAY LIST

Suppose we initialize an array list with an empty array of length 1.

We then add an element.



ADDING N ELEMENTS TO AN ARRAY LIST

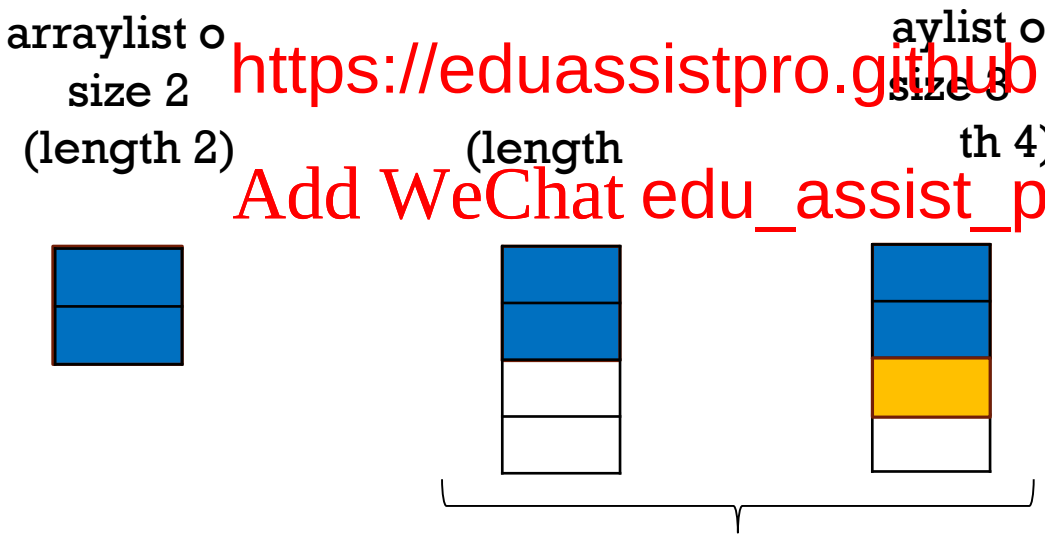
Suppose each time we add to a full array list, we double the length of the array.

arraylist of size 1 (length 1)

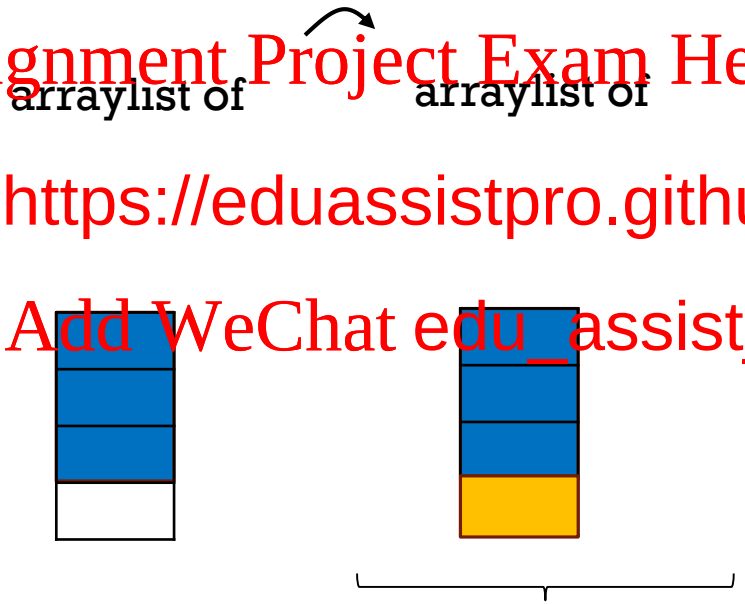
(length 2)

add second element

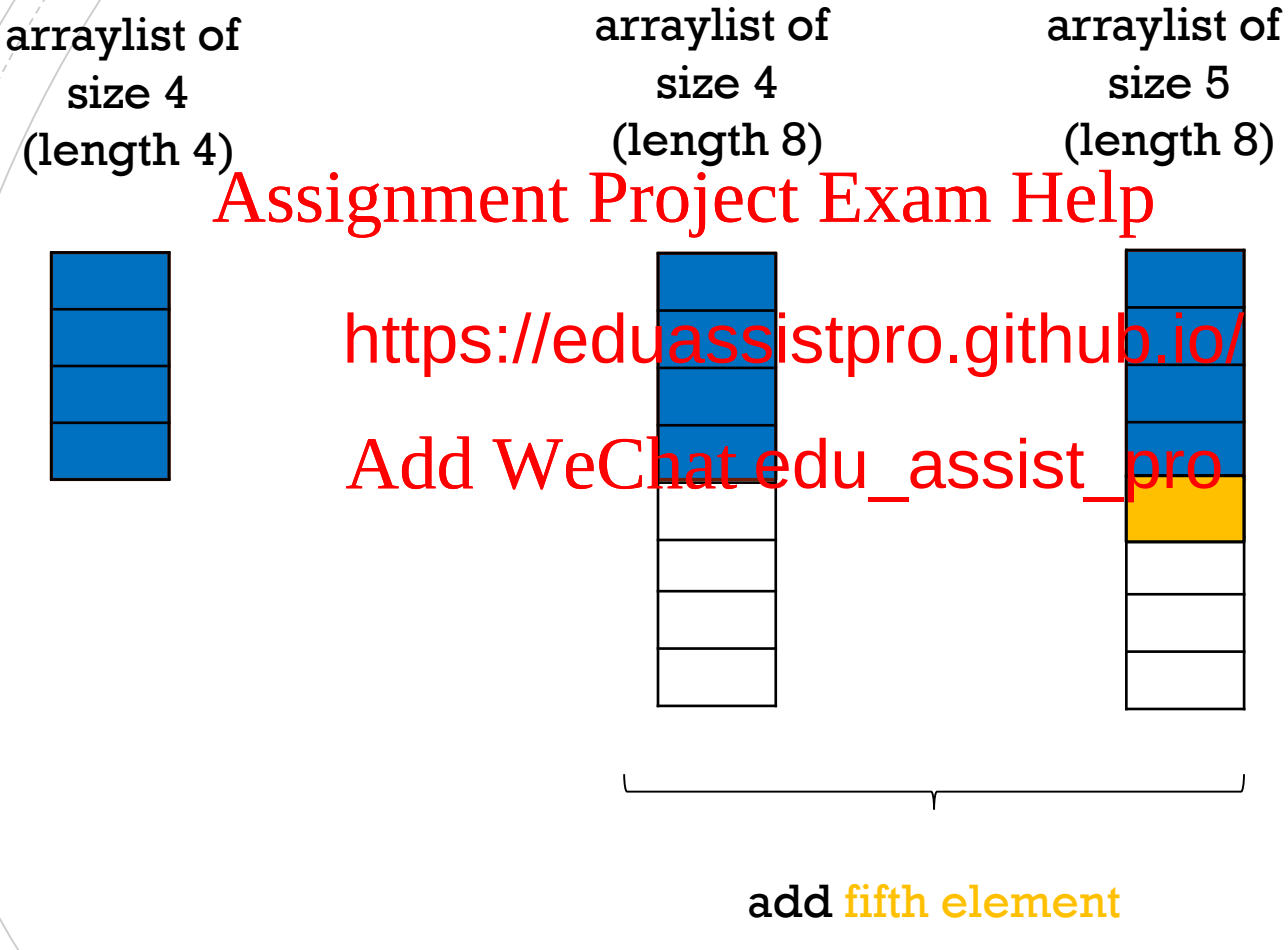
ADDING N ELEMENTS TO AN ARRAY LIST



ADDING N ELEMENTS TO AN ARRAY LIST

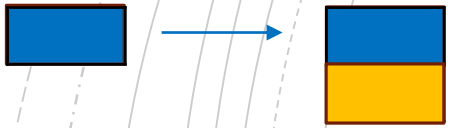


ADDING N ELEMENTS TO AN ARRAY LIST



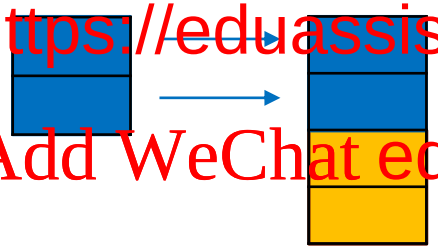
ADDING N ELEMENTS TO AN ARRAY LIST

Double length and copy one element



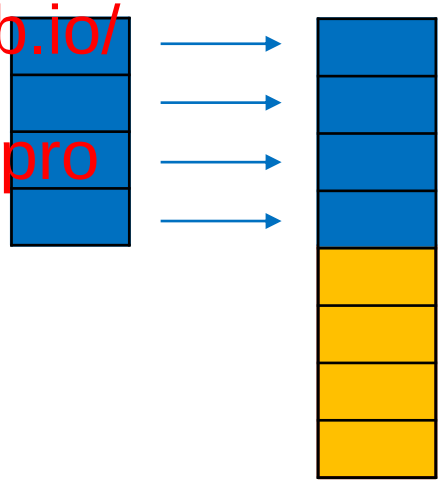
add two elements

Double length and copy two elements



add 3-4 elements

Double length and copy four elements



add 5-8 elements

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

A:

Q: How many cop uired to add N elements to an empty arra

A:

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

A: $2 \cdot 2 \cdots 2 = 2^k = N \rightarrow k = \log_2 N$

Q: How many cop uired to add N elements to an empty arra

A:

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

A: $2 \cdot 2 \cdots 2 = 2^k = N \rightarrow k = \log_2 N$

Q: How many cop uired to add N elements to an empty arra

A: $1 + 2 + 4 + \ldots + 2^{k-1}$

SERIES

- *Geometric series*

$$\sum_{i=0}^{n} x^i = 1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x}$$

- *Arithmetic series*

$$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{1}{2} n(n+1)$$

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

A: $2 \cdot 2 \cdots 2 = 2^k = N \rightarrow k = \log_2 N$

Q: How many cop uired to add N elements to an empty arra

A: $1 + 2 + 4 + \ldots + 2^{k-1} = \sum_{i=0}^{k-1} 2^i = \frac{1-2^k}{1-2} = 2^k - 1 = N - 1$

JAVA ARRAYLIST CLASS

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

- It uses an array as the und
- It grows the array (by 50%, not 100%) wh is full and a new element is added.
- As a client, we don't have access to the fields directly. We need to use the methods provided (get(), set(), …) to manipulate the list.

JAVA ARRAYLIST – GENERIC CLASS

- `ArrayList` **is a generic class with a type parameter.**
- **When you create an object of type** `ArrayList` **you specify the type of the element** **pending to** `ArrayList` **a class name enclosed** **rackets.**
- **Example:**

If we write `int` instead we get a compile-time error!

```
// creates an arraylist of integers with initial capacity 10
ArrayList<Integer> words = new ArrayList<Integer>();

// creates an arraylist of shapes with initial capacity 23
ArrayList<Shape> myShapes = new ArrayList<Shape>(23);
```

WRAPPER CLASSES

- `Integer`, `Double`, and `Character` wrap a value of the primitive type `int`, `double`, and `char` (respectively) in an object. Thus, they turn primitive types into reference types.

- The conversion between their wrappers is done automatically. For example, the followi t cause a compile-time error:

```
Integer x = 5;
```

- Note that these classes have static methods/attributes that you might have already used. For example: `Integer.MAX_VALUE`, `Double.parseDouble()`

AUTOBOXING AND UNBOXING

- *Autoboxing* is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example converting an int to an Integer.

- If the conversion goes the other way, this is called *unboxing*.

```
Integer x = new Integer(5);
int y = x;
```

- https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html

IMMUTABLE TYPES

- Note that `Integer`, `Double`, and `Character` are **immutable r** ing).

- As with `String`, you can ap ates values, but you are never changing the actual Object. A new Object gets created each time we "change" a value.

WHY WRAPPER CLASSES?

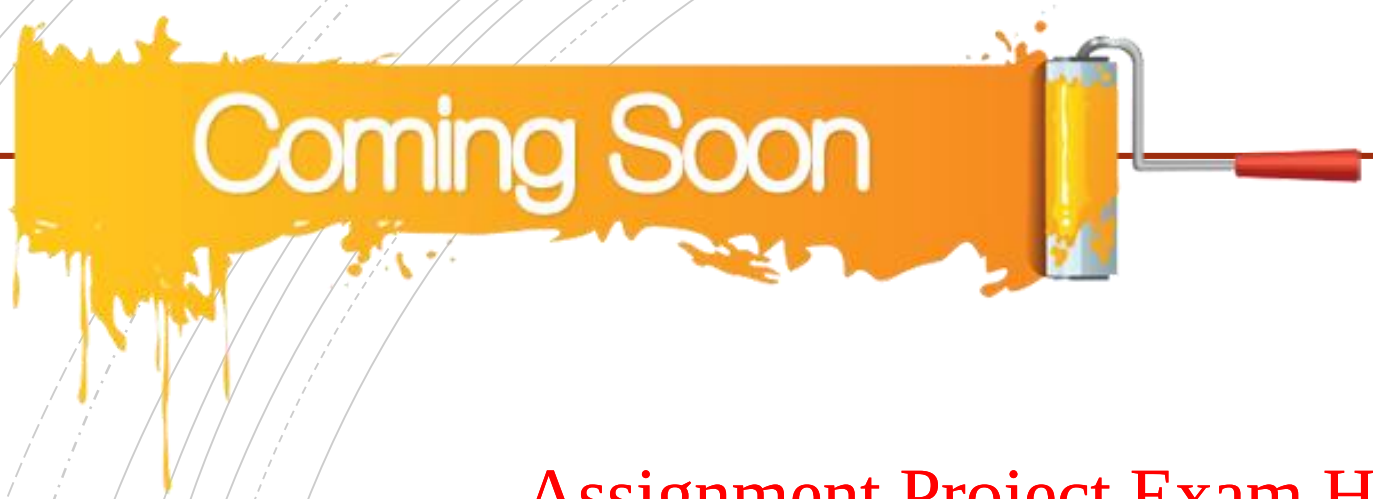
It is much simpler (in terms of code re-use) to have `ArrayList` req bject (a reference type), instead of using primitiv

For example, all reference types can be compared using `.equals()`, while we have to use `==` for primitive types.

THE FOREACH LOOP

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

for(i                          {
   System.out.pri              nt);
}
```


The foreach loop (also called enhanced for loop) can make your code more readable and can be convenient to use. It is not helpful when you need to refer to the index of an element.



In the next

- Linked li