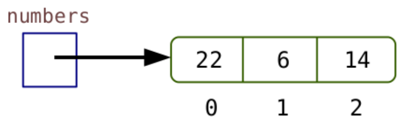
Model 1 Array Lists

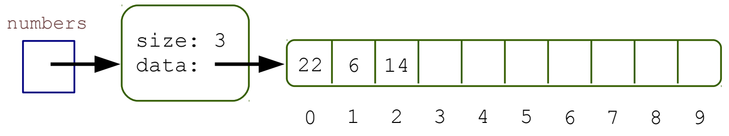
Arrays store elements in one ***contiguous*** block of memory. Since elements are stored together, you can immediately access any element by its index.

```
int[] numbers = {22, 6, 14};
System.out.println(numbers[1]);
```



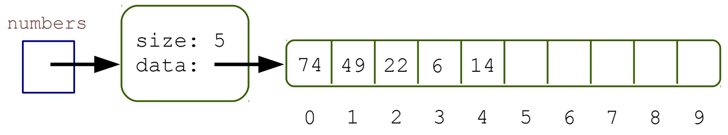
The `ArrayList` collection implements `List` and uses an array (internally) to store its elements.

```
ArrayList<Integer> numbers = new ArrayList<>();
numbers.add(22);
numbers.add(6);
numbers.add(14);
```



When new values are inserted, existing array elements are moved to the right.

```
numbers.add(0, 49);
numbers.add(0, 74);
```



If the array fills up, `ArrayList` automatically creates a new array about 50% larger. All current values must be copied into the new array, and the old array is then garbage collected.

Questions (15 min)

Start time:

1. Why does Java use the name `ArrayList`? (What do the words `Array` and `List` indicate?)

The `ArrayList` collection implements the `List` interface using an array. In general, the first word tells how it's implemented, and the second word is the interface.

2. How many array operations (i.e., integer assignments) were required to add 49 and 79 to the front of the second diagram in Model 1?

There are 9 total array operations: adding 49 takes 4 operations (3 shifts + 1 add), and adding 74 takes 5 operations (4 shifts + 1 add).

3. Imagine the internal array for `numbers` is full (i.e., with size=10 above). If you request one more element to be added (at the end), how big will the new array be?

50% of 10 is 5, so the new array will be 15 elements.

4. Continuing the previous question, what operations are required to add one more element when the array is full? Briefly describe each operation, beginning with creating the new array.

First, a new array of length 15 must be created. Then, the 10 elements must be copied from the old array to the new array. Next, the new element is added to the end and the `size` attribute is incremented. Finally, the old array is garbage collected.

5. Discuss why `ArrayList` is a poor choice of `List` in the program below:

```java
import java.util.ArrayList;
import java.util.List;

public class ArraysAreBad {

    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        System.out.println("Start");
        addAndRemove(list);
        System.out.println("Done!");
    }

    public static void addAndRemove(List<String> list) {
        for (int i = 0; i < 1000000; i++) {
            list.add(0, "A");  // add at index 0
        }
        for (int i = 0; i < 1000000; i++) {
            list.remove(0);  // remove at index 0
        }
    }
}
```

Each insertion at the beginning of the array takes n operations (where n is the current size of the collection) in order to shift existing elements down.

6. Arrays are simple and effective. Why would we want anything but ArrayList?

Arrays are inefficient when adding and removing items from the beginning (or the middle).