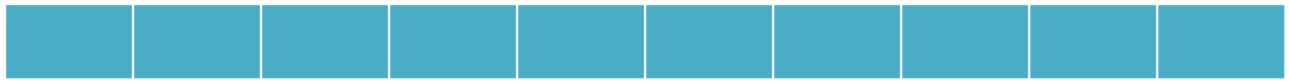
Creating your own ArrayList

Reinventing the wheel helps you appreciate the wheel.

ArrayList under the hood

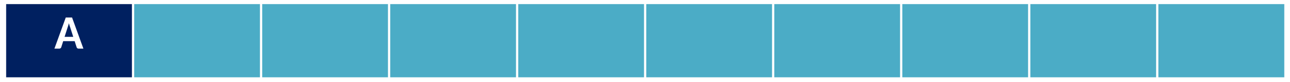
- The real ArrayList has a data field that is just a regular array, and an int to keep track of the # of elements added into the list (logical size).
- The actual size of the array will hold some buffer space. If the buffer space runs out, the array size is increased to double its previous size.
- If items are removed and the logical size is less than 1/3 the actual size, the buffer space is cut in half.



numElements

```
List<String>words = new ArrayList<String>();
```

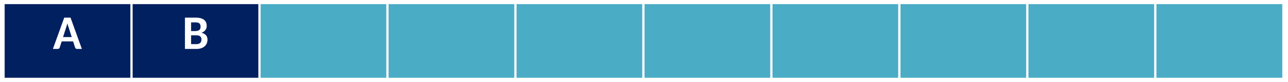
0



<u>numElements</u>

1

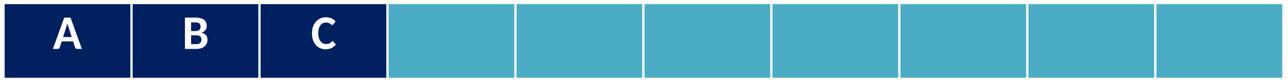
```
List<String>words = new ArrayList<String>();
words.add(“A”);
```



<u>numElements</u>

2

```
List<String>words = new ArrayList<String>();
words.add("A");
words.add("B");
```



numElements

3

```
List<String>words = new ArrayList<String>();
words.add("A");
words.add("B");
words.add("C");
```

A	B	C	D	E	F	G	H	I	J

numElements

10

```
List<String>words = new ArrayList<String>();
words.add("A");
words.add("B");
words.add("C");
//code to add D,E,F,G,H,I,J
```

A	B	C	D	E	F	G	H	I	J	K			....

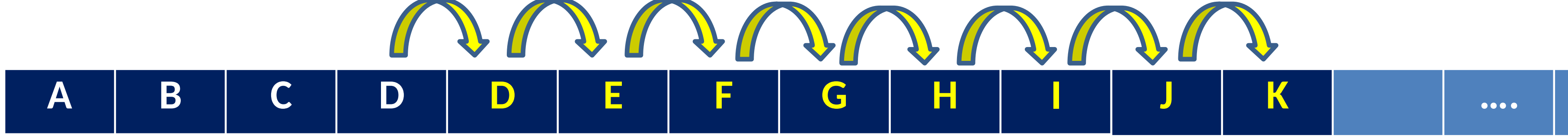
```
List<String>words = new ArrayList<String>();
words.add("A");
words.add("B");
words.add("C");
//code to add D,E,F,G,H,I,J
words.add("K");          //buffer size increased to 20
```

11

A	B	C	D	E	F	G	H	I	J	K			....

11

```
List<String>words = new ArrayList<String>();
words.add(“A”);
words.add(“B”);
words.add(“C”);
//code to add D,E,F,G,H,I,J
words.add(“K”);          //buffer size increased to 20
words.add(3,”X”);
```

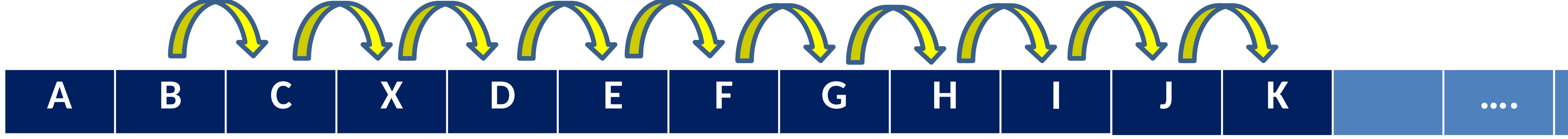


11

```
List<String>words = new ArrayList<String>();
words.add("A");
words.add("B");
words.add("C");
//code to add D,E,F,G,H,I,J
words.add("K");          //buffer size increased to 20
words.add(3,"X");        //elements from index 3 to the end of logical
                         //size are shifted one space to the right
```

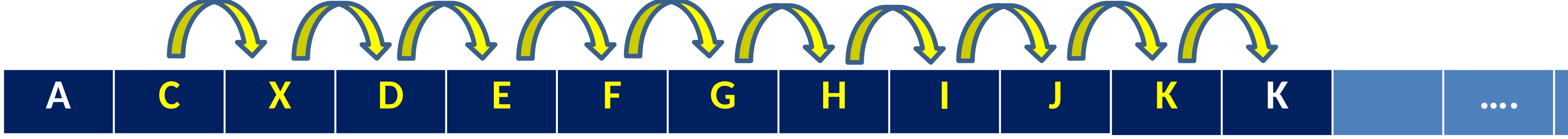
A	B	C	X	D	E	F	G	H	I	J	K		....

```
List<String>words = new ArrayList<String>();                12
words.add(“A”);
words.add(“B”);
words.add(“C”);
//code to add D,E,F,G,H,I,J
words.add(“K”);          //buffer size increased to 20
words.add(3,”X”);        //elements from index 3 to the end of logical
                         //size are shifted one space to the right,
                         //then new element is copied in and size is
                         //increased by one.
```



12

```
List<String>words = new ArrayList<String>();
words.add("A");
words.add("B");
words.add("C");
//code to add D,E,F,G,H,I,J
words.add("K");         //buffer size increased to 20
words.add(3,"X");       //elements from index 3 to the end of logical
                        //size are shifted one space to the right,
                        //then new element is copied in and size is
                        //increased by one.
words.remove(1);
```



12

```
List<String>words = new ArrayList<String>();
words.add(“A”);
words.add(“B”);
words.add(“C”);
//code to add D,E,F,G,H,I,J
words.add(“K”);          //buffer size increased to 20
words.add(3,”X”);        //elements from index 3 to the end of logical
                         //size are shifted one space to the right,
                         //then new element is copied in and size is
                         //increased by one.
words.remove(1);         //elements from index 2 to the end of logical
                         //size are shifted one space to the left,
```

A	C	X	D	E	F	G	H	I	J	K	K		....

```
List<String>words = new ArrayList<String>();                 11
words.add("A");
words.add("B");
words.add("C");
//code to add D,E,F,G,H,I,J
words.add("K");          //buffer size increased to 20
words.add(3,"X");        //elements from index 3 to the end of logical
                         //size are shifted one space to the right,
                         //then new element is copied in and size is
                         //increased by one.
words.remove(1);         //elements from index 2 to the end of logical
                         //size are shifted one space to the left,
                         //then logical size is decreased by one.
```

The Interface

- Your MyArrayList must implement the following shorter version of the List interface:

```
public interface ListInterface<anyType>
{
    public boolean add(anyType x);
    public boolean add(int index, anyType x);
    public int size();
    public anyType set(int index, anyType x);
    public anyType get(int index);
    public anyType remove(int index);
}
```

- anyType means that you can send it an Object of… any type.

```
public class MyArrayList<anyType> implements ListInterface<anyType>
{
    private Object[] list;                         //the actual container
    private int numElements;                       //keep track of logical size

    public MyArrayList()
    {
      list = new Object[10];                       //start with buffer size of 10
      numElements = 0;                             //and zero logical elements
    }
   //more methods here
}
```

- Implementing ListInterface means you must define all interface methods concretely.
- Complete helper methods doubleCapacity() and cutCapacity() to adjust the buffer size when necessary.