# Tips and Common Points of Confusion

# How to Get Started

Stuck on how to even get started on LinkedListDeque and/or ArrayDeque? One great first step is implementing SLList and/or AList. Starter code can be found in:

- SLList: https://github.com/Berkeley-CS61B/lectureCode-fa22/blob/main/lec5_lists2/DIY/SLList.java
- AList: https://github.com/Berkeley-CS61B/lectureCode-fa22/blob/main/lec7_lists4/DIY/AList.java

To make this more time efficient:

- Work with a friend or two or three.
- See solutions (SLList and AList) (for when you get stuck or just want to compare).

# How to Get Started (Part 2)

Try implementing just the empty Deque, and compare your data structure to the idea developed in class.

- e.g. your LinkedListDeque() constructor should probably build either:
  - The top figure from this slide (if you're doing two sentinel nodes).
  - The top figure from this slide (if you're going circular).

Compare the output of your code using the visualizer.

- You can use the visualizer from inside IntelliJ. See the 61B Plugin guide.

# Other General Tips

Take things a little at a time.

- Writing tons of code all at once is going to lead to misery and only misery.
- If you wrote too much stuff and feel overwhelmed, comment out whatever is unnecessary.

If your first try goes badly, don't be afraid to scrap your code and start over.

- The amount of code for each class isn't actually that much (my solution is about 130 lines for each .java file, including all comments and whitespace).

Consider not doing resizing at all until you know your code works without it.

- Resizing is a performance optimization (and is required for full credit).

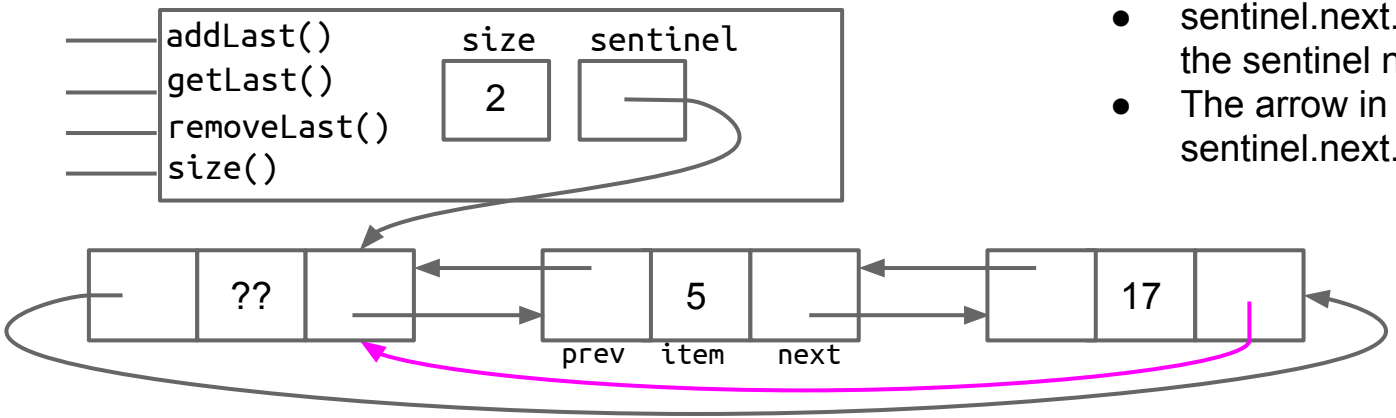
# Common Point of Confusion

Any time I drew an arrow in class that pointed at an object, the pointer was to the ENTIRE object, not a particular field of an object.

- For example, the bits in the “sentinel” box do not point at the “next” field of the leftmost node. They instead point to the ENTIRE node.
- (in fact it is impossible for a reference to point to the fields of an object in Java)

Examples:

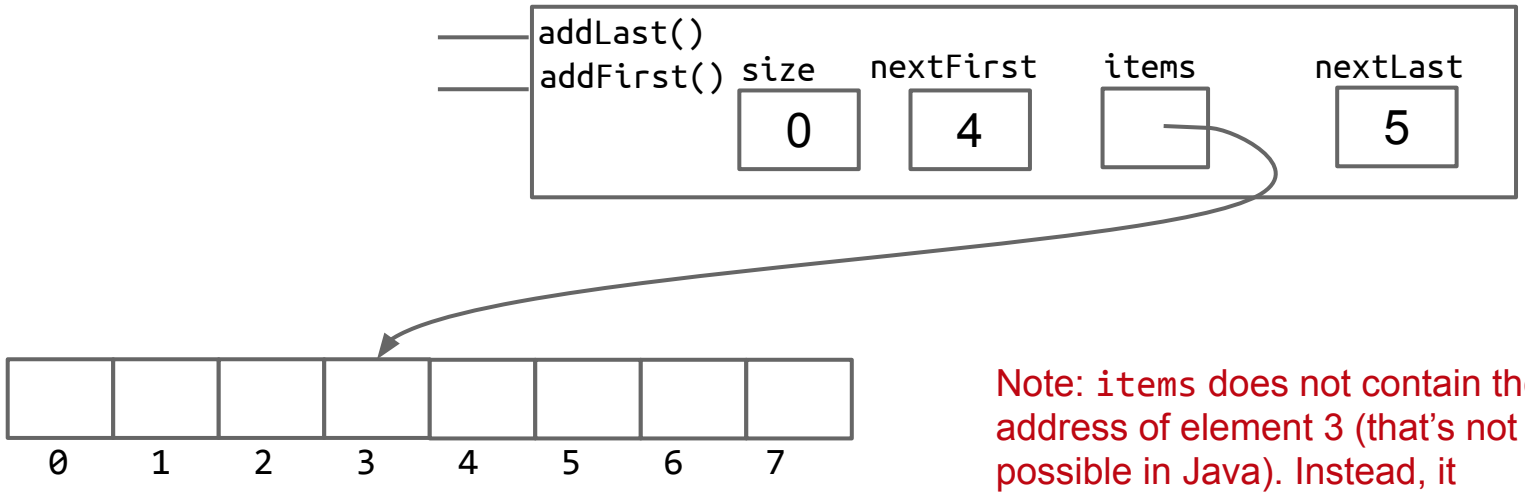
- sentinel.next.next is the node with item=17.
- sentinel.next.next.next is the sentinel node.
- The arrow in purple is sentinel.next.next.next



# Circular ArrayDeque Demo

# ArrayDeque Tips

Starting from an empty ArrayDeque: Conceptual Deque: []



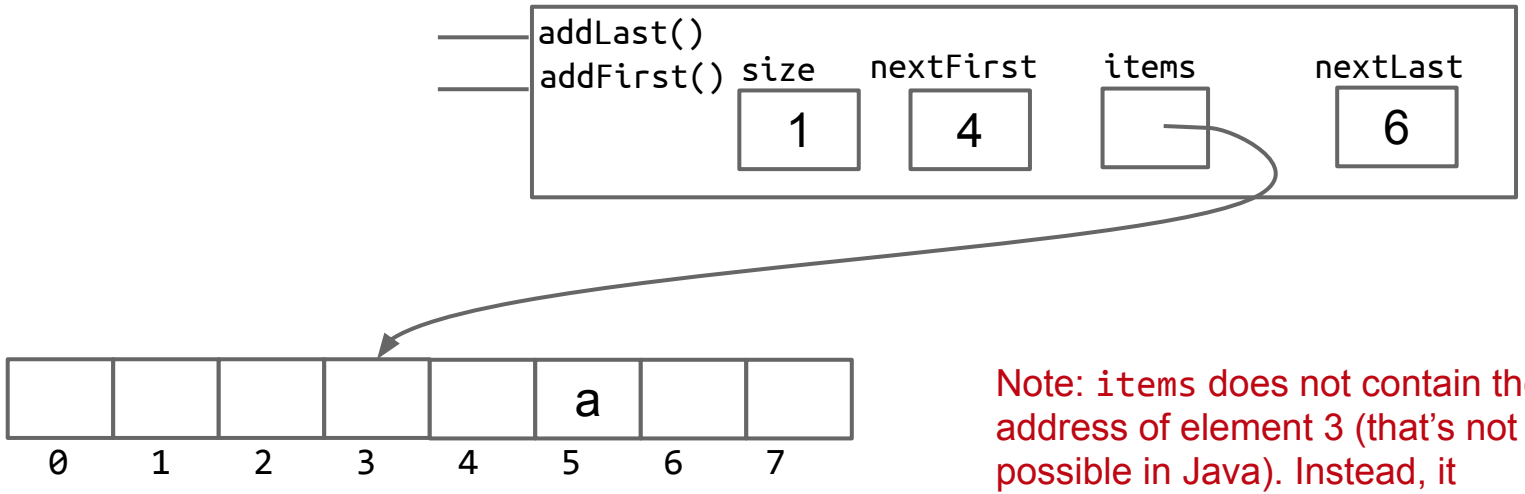

Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque: Conceptual Deque: [a]

- addLast("a")



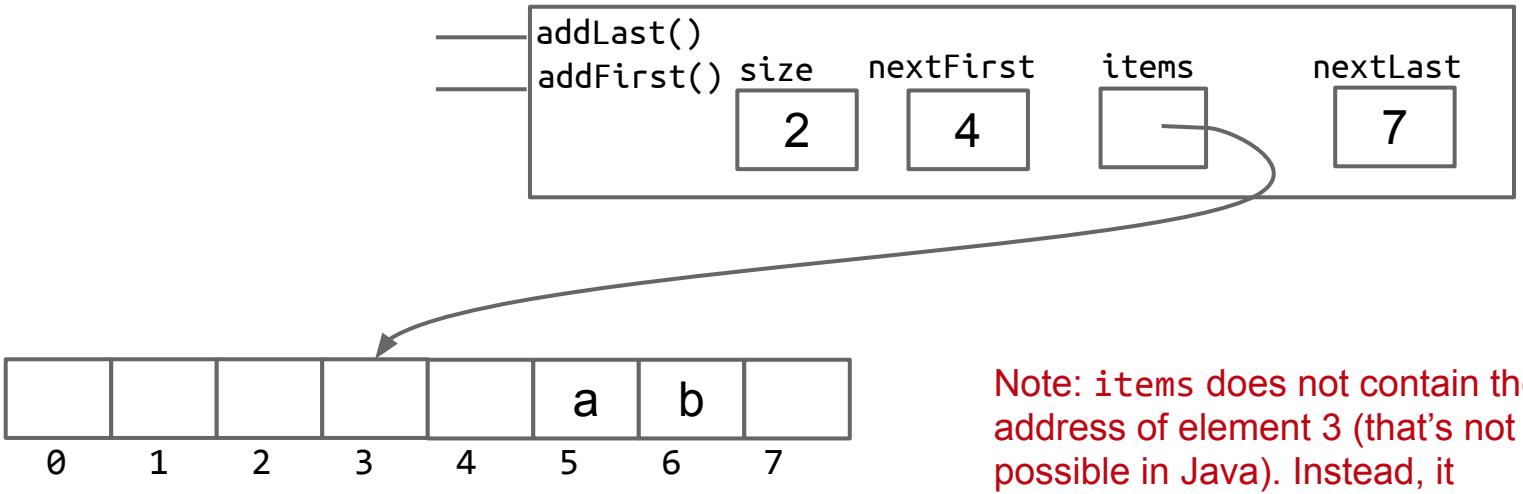

Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque: Conceptual Deque: [a, b]

- addLast("a")
- addLast("b")




Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

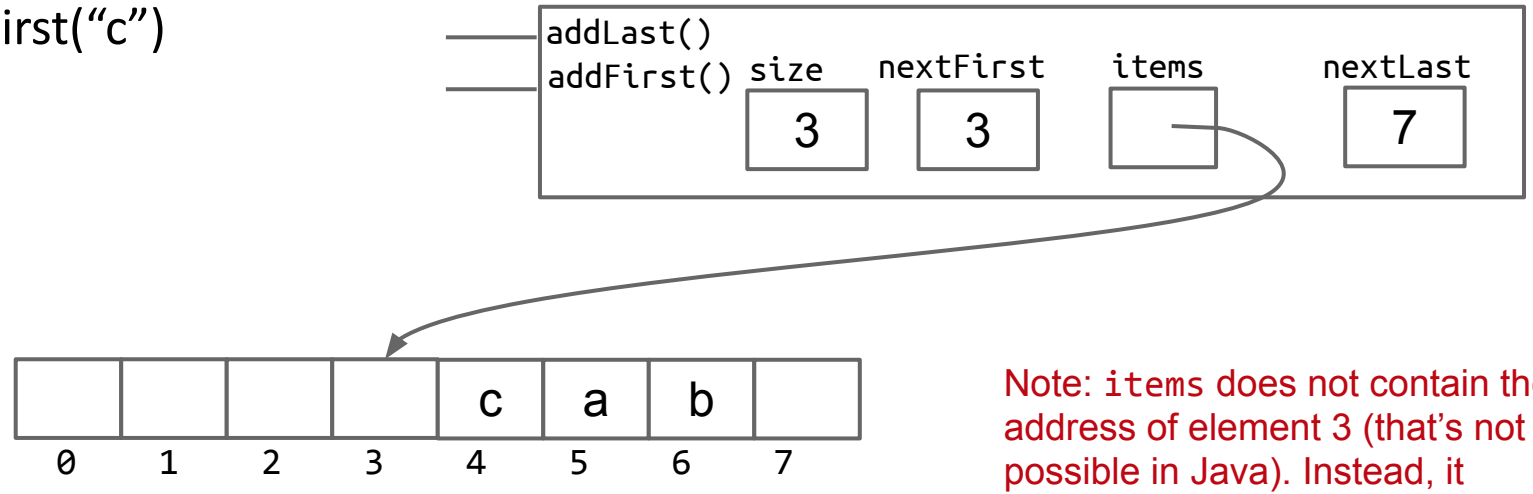
You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque:

Conceptual Deque: [c, a, b]

- addLast("a")
- addLast("b")
- addFirst("c")




Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

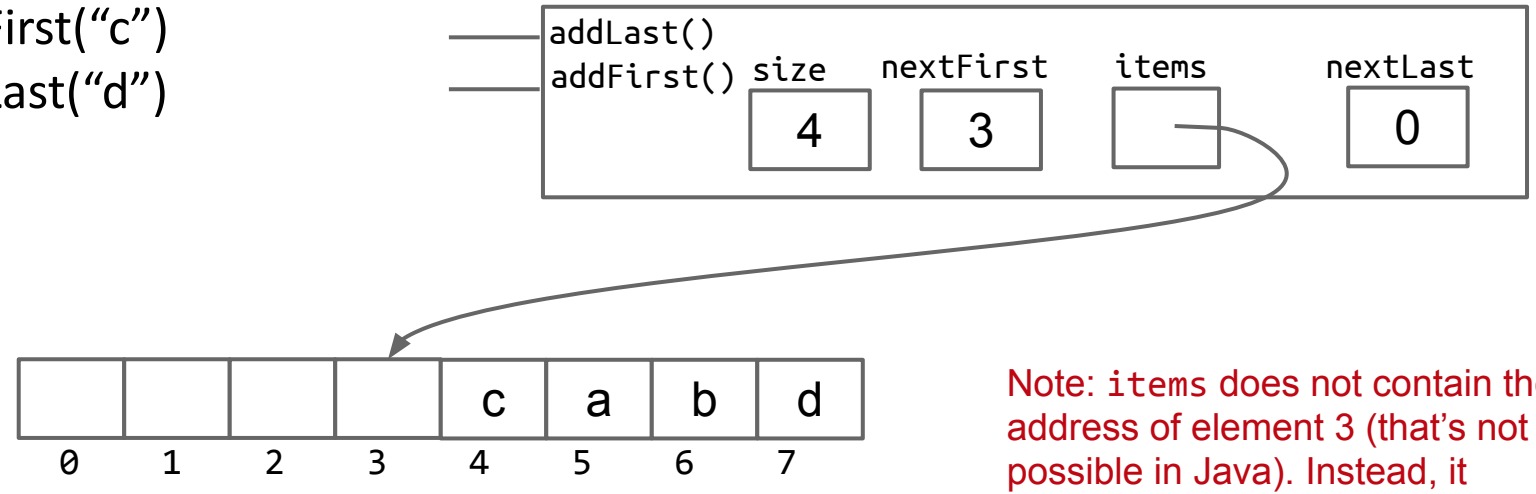
You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque:

Conceptual Deque: [c, a, b, d]

- addLast("a")
- addLast("b")
- addFirst("c")
- addLast("d")




Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

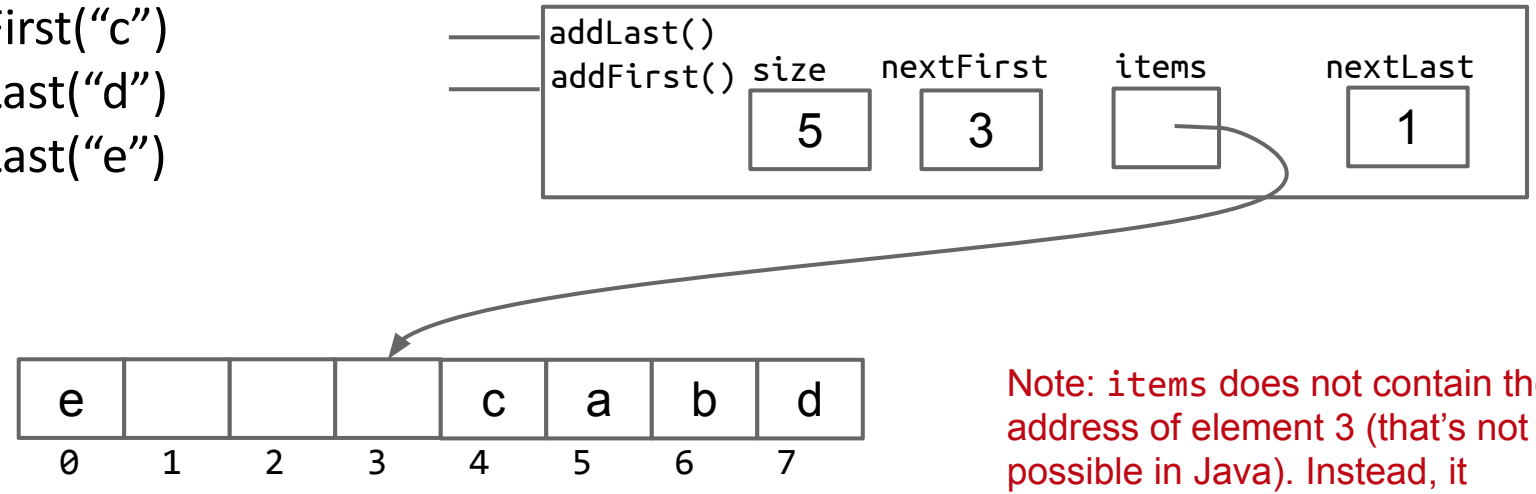
You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque:

Conceptual Deque: [c, a, b, d, e]

- addLast("a")
- addLast("b")
- addFirst("c")
- addLast("d")
- addLast("e")




Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

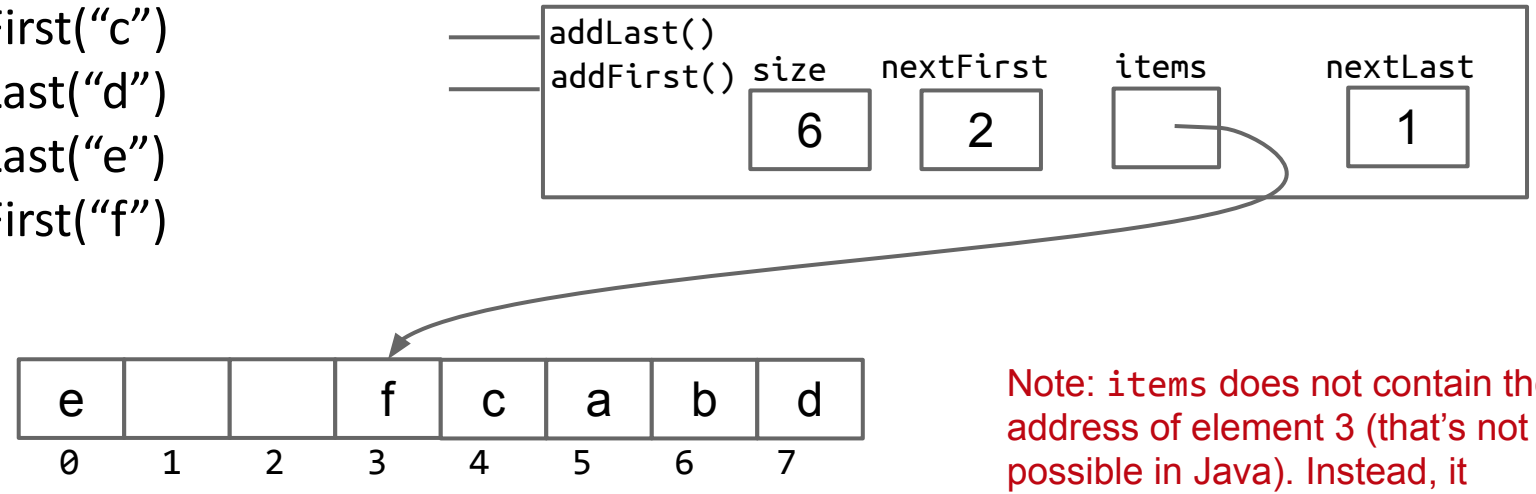
You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque:

Conceptual Deque: [f, c, a, b, d, e]

- addLast("a")
- addLast("b")
- addFirst("c")
- addLast("d")
- addLast("e")
- addFirst("f")




Note: `items` does not contain the address of element 3 (that's not possible in Java). Instead, it contains the address of the entire array!

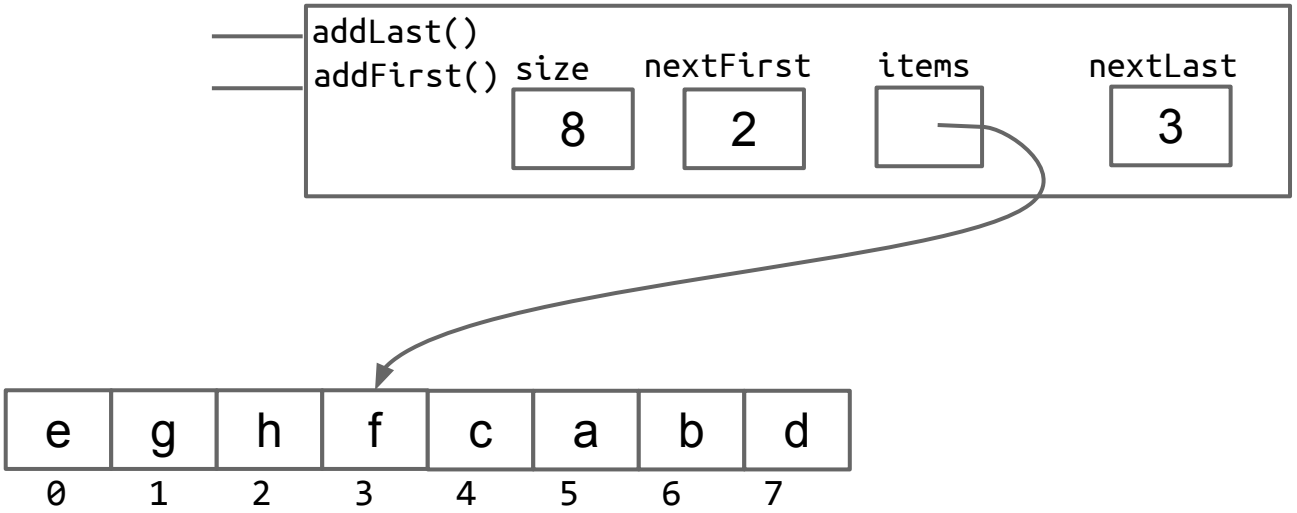
You don't have to start your array at 4 or 5. I just picked these arbitrarily.

# ArrayDeque Tips

Starting from an empty ArrayDeque:

Conceptual Deque: [f, c, a, b, d, e, g, h]

- addLast("a")
- addLast("b")
- addFirst("c")
- addLast("d")
- addLast("e")
- addFirst("f")
- addLast("g")
- addLast("h")




After the next addLast operation, we'll need a bigger array.

# Resizing: You Can Pick Whatever Representation You Prefer

Starting from an empty ArrayDeque:

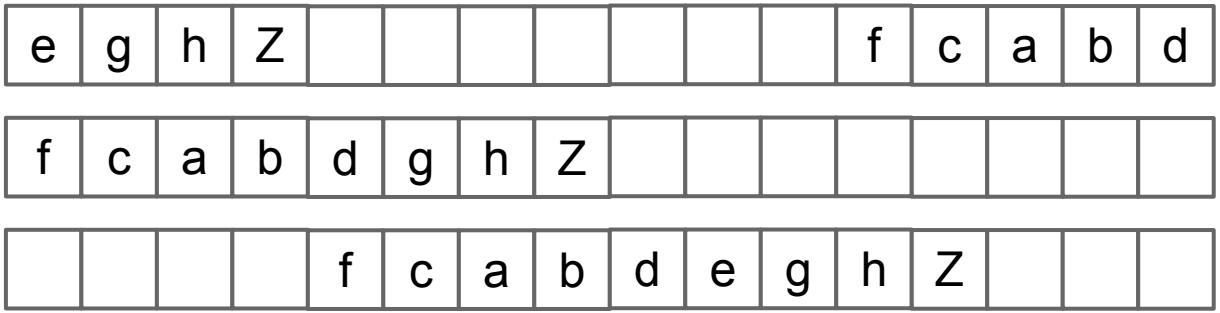
Conceptual Deque: [f, c, a, b, d, e, g, h, Z]

- addLast("a")
- addLast("b")
- addFirst("c")
- addLast("d")
- addLast("e")
- addFirst("f")
- addLast("g")
- addLast("h")
- **addLast("Z")**

Before addLast("Z"):

| e | g | h | f | c | a | b | d |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

After addLast("Z"): (all of the below are valid choices)




It's your choice what array to use, e.g. all of the three ideas above are valid!

- Note: You don't have to use arraycopy if a for loop is easier for your choice.
- Recall Plato's Allegory of the Cave from lecture.

# Writing .iterator() and Equals

Lecture 11 shows how to write these methods. See:

https://docs.google.com/presentation/d/1lIR4--P9NrBqRL9xqP_RQYyK1WJBrBEbriLVpatrRqk/edit

- **Especially important: See slide on instanceof.**

Or code developed in lecture 11:

- https://github.com/Berkeley-CS61B/lectureCode-fa22/blob/main/lec11_inheritance4/ArraySet.java

# Why am I getting T, expected type T in Iterator?

Make sure your inner class looks like:

- private class DequeIterator implements Iterator<T> {

And that it does not look like:

- private class DequeIterator<T> implements Iterator<T> {