ADT & Implementation
Queues

COMP128 Data Structures

Different names, same behavior

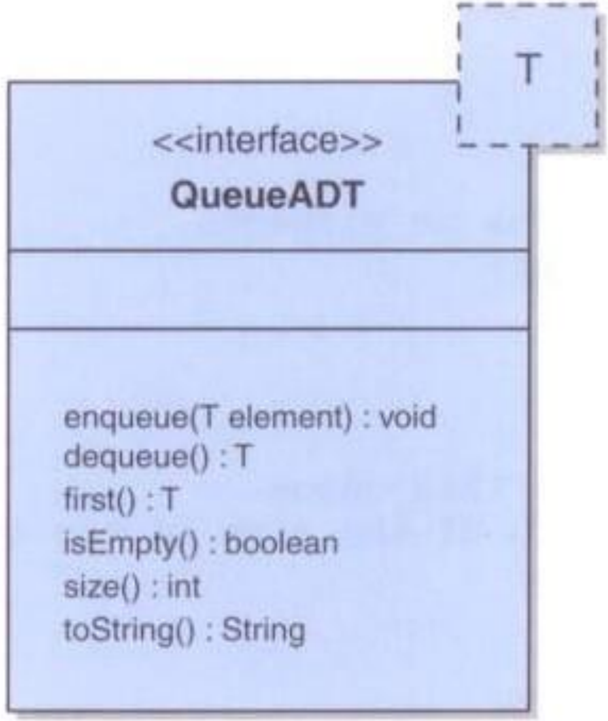
Textbook Queue ADT	Java Queue
enqueue	add, offer
dequeue	remove, poll
first	element, peek
isEmpty	isEmpty
size	size

Be aware of the difference between these!

Queue ADT

Defining our own interface for a queue, using only the classic operations

** See today's activity for implementation*



Queues often use linked-node list structures

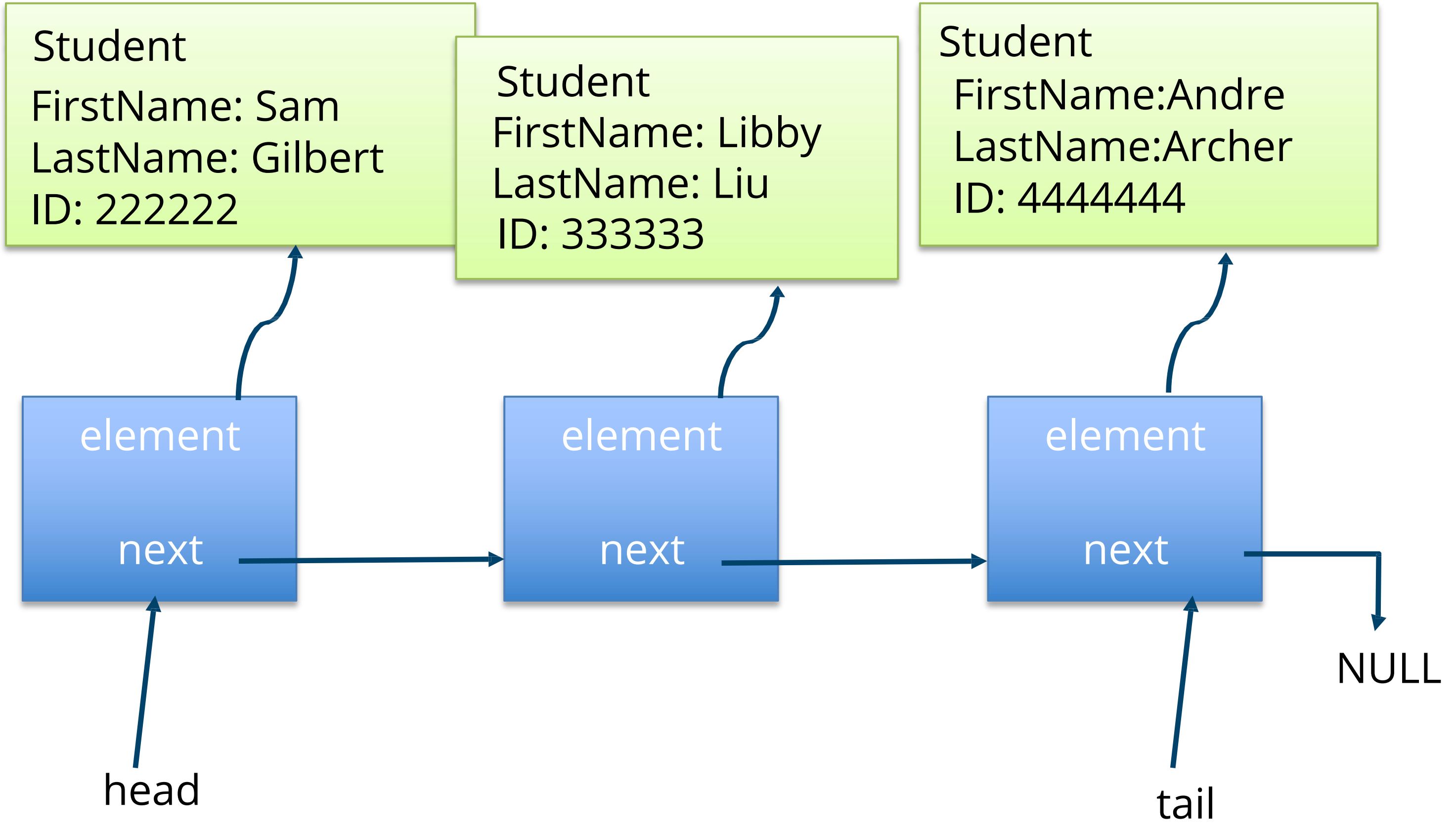
- Array can be used, in a circular fashion
- Concentrate on linked version

Linked Structures use a Node class

```
/** LinearNode represents a node in a
linked list.
 * @author Java Foundations
 * @version 4.0
 */
public class LinearNode<E> {
    private LinearNode<E>
    next;  private E element ;
    /**
     * Creates an empty node.
     */
    public LinearNode()
    {
     next = null;
     element = null;
    }
```

```
/* Creates a node storing the
specified element.
* @param elem the element to be
stored  within the new node
*/

public LinearNode(E elem){
    next = null;
    element =
    elem;
}
… getters and setters
```

element
next

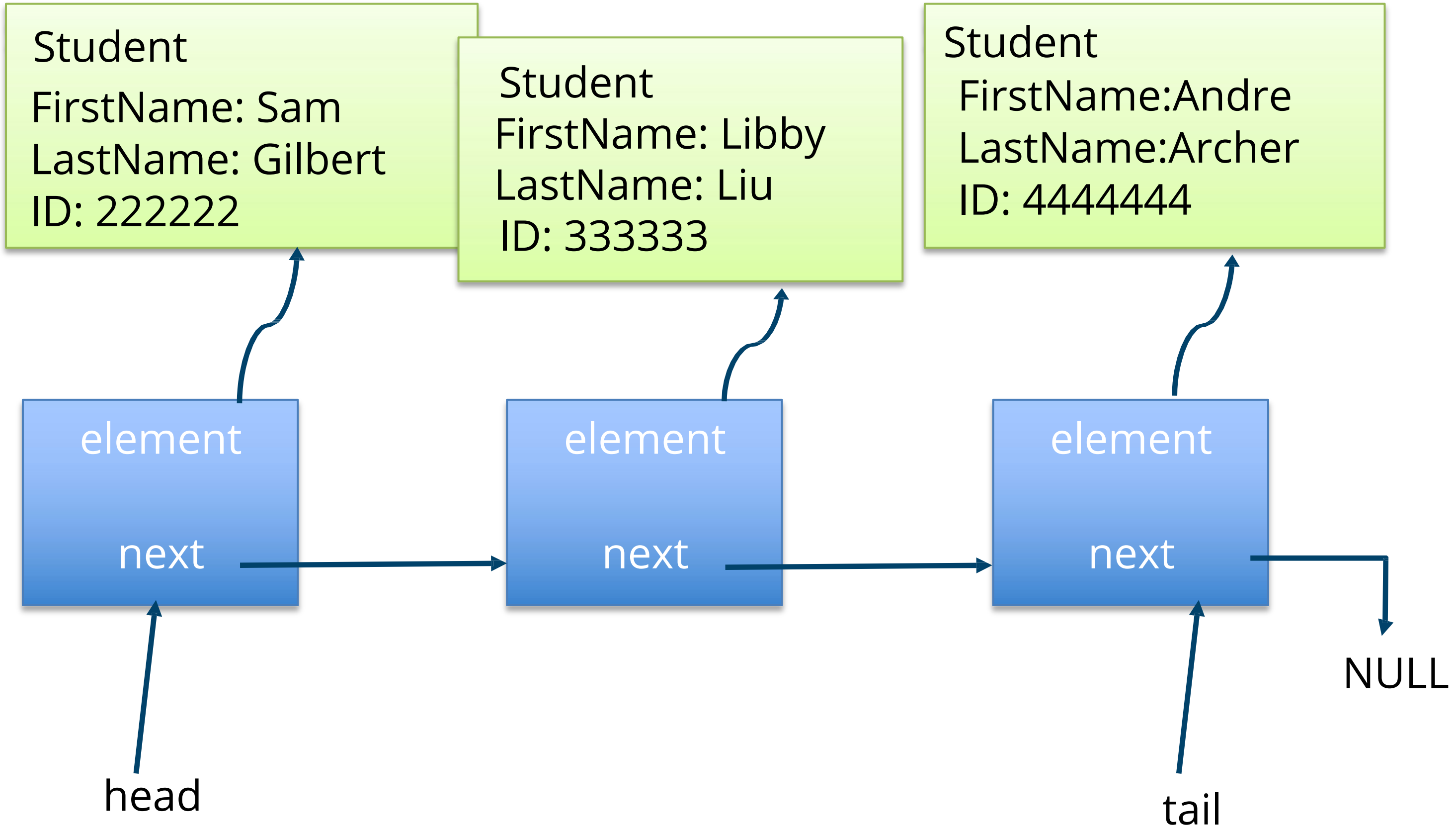


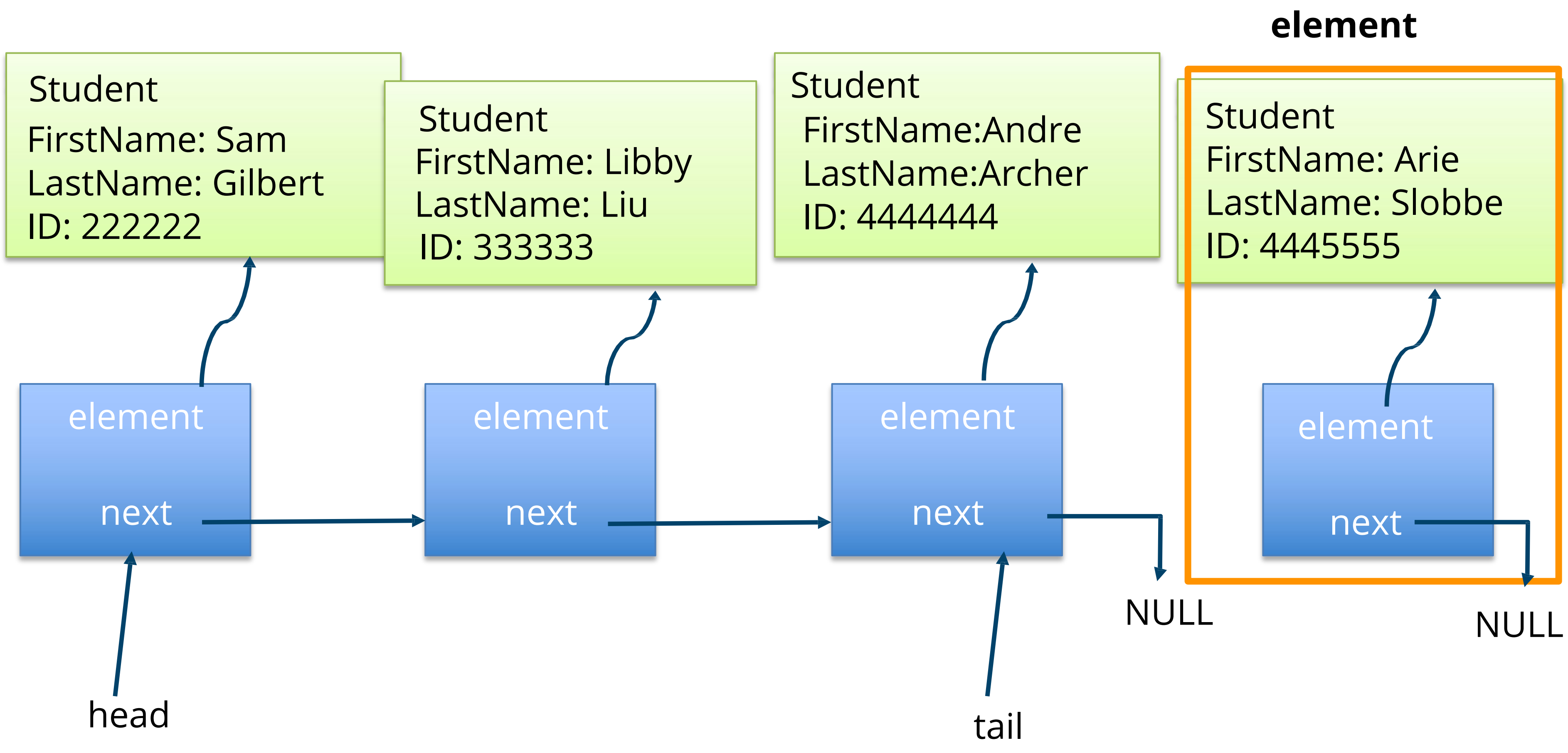
The bare minimum Queue class contains a declaration of the LinearNode class and private instance variables of type Node for the head and the tail, along with methods for all of the operations.

The bare minimum Queue class contains a declaration of the LinearNode class and private instance variables of type Node for the head and the tail, along with methods for all of the operations.

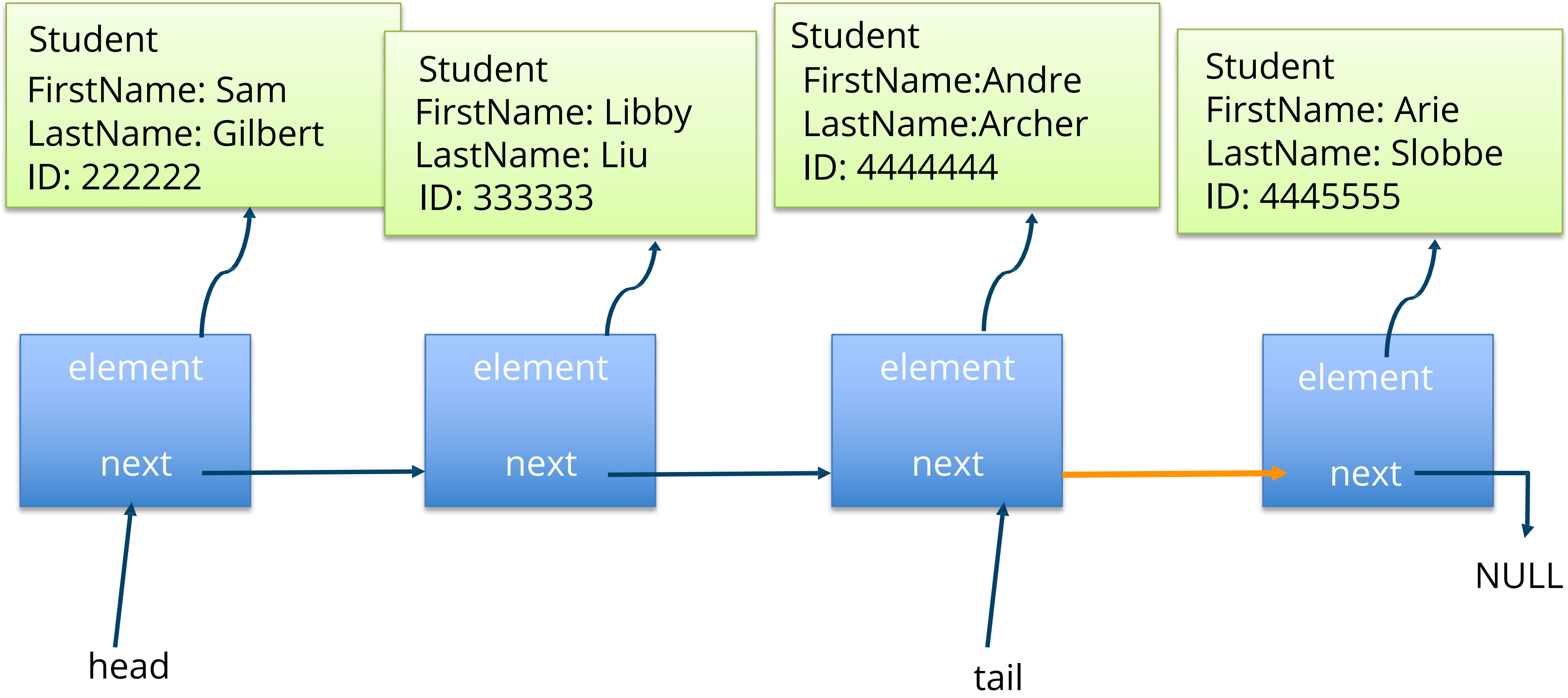
```
/*LinkedQueue represents a linked implementation of a queue.
 * @author Java Foundations
 * @version 4.0
 */
public class LinkedQueue<T> implements QueueADT<T> {
 private int count;
 private LinearNode<T> head, tail;

 /* Creates an empty queue. */

 public LinkedQueue(){
  count = 0;
  head = tail = null;
 }
    …   QueueADT methods
```

void enqueue(T element)

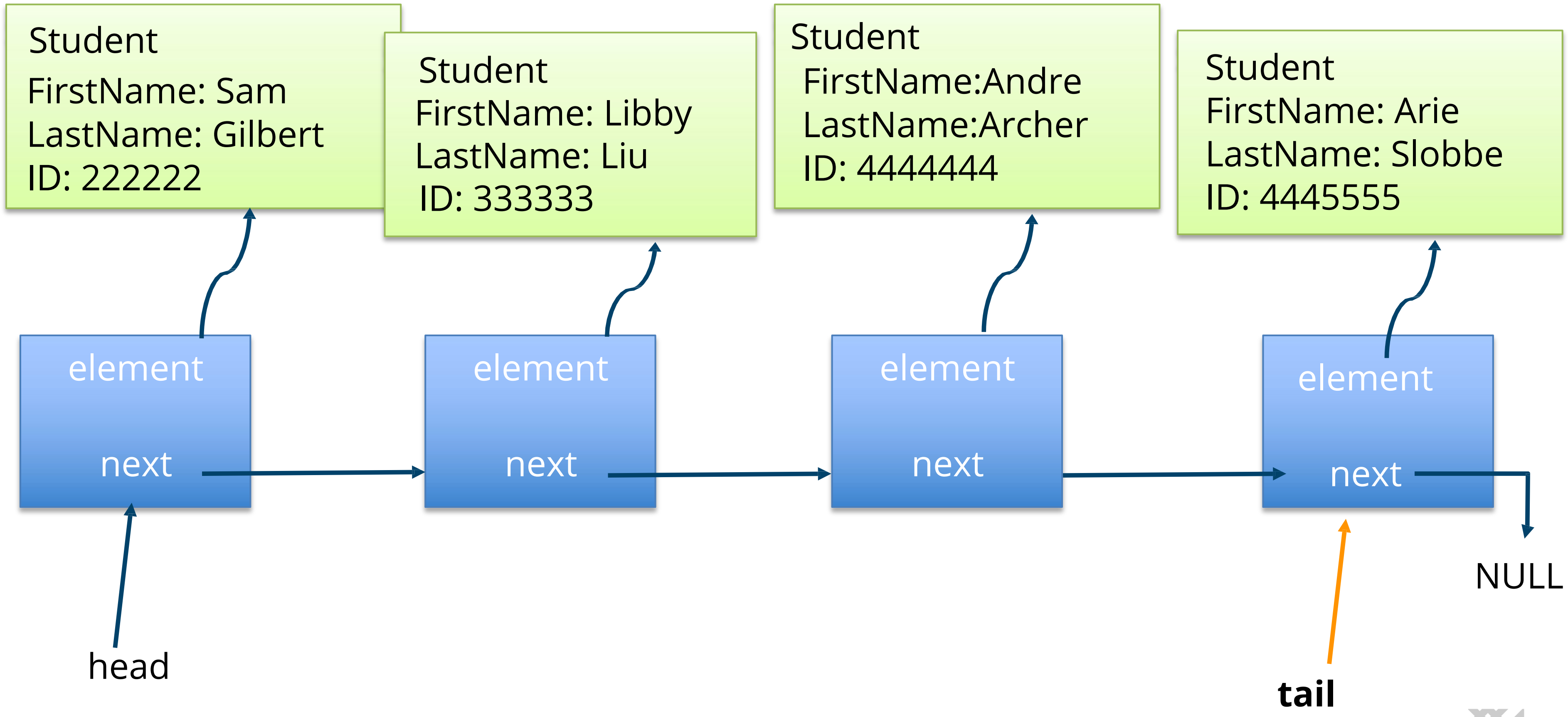




Step 1: Create new node for the element



Step 2: Set next for the current tail element



Step 3: Update tail to the added node

Algorithm Summary

1. element is the parameter object passed in
2. LinearNode newNode = new LinearNode();
3. newNode.element = element;
4. tail.next = newNode;
5. tail = newNode;

At worst, how long do these take?

- `boolean enqueue(T element)`
  - Appends the specified element to the end of this list.
- `T dequeue()`
  - Removes the first element from this list, if it is present.
- `T first()`
  - Returns but does not remove the first element from this list, if it is present.
- `boolean isEmpty()`
  - returns true if the queue is empty.
- `int size()`
  - Returns the number of elements in the queue
- `String toString()`
  - string representation of the queue

At worst, how long do these take?

O(1)

- `boolean enqueue(T element)`
  - Appends the specified element to the end of this list.
- `T dequeue()`
  - Removes the first element from this list, if it is present.
- `T first()`
  - Returns but does not remove the first element from this list, if it is present.
- `boolean isEmpty()`
  - returns true if the queue is empty.
- `int size()`
  - Returns the number of elements in the queue
- `String toString()`
  - string representation of the queue

At worst, how long do these take?

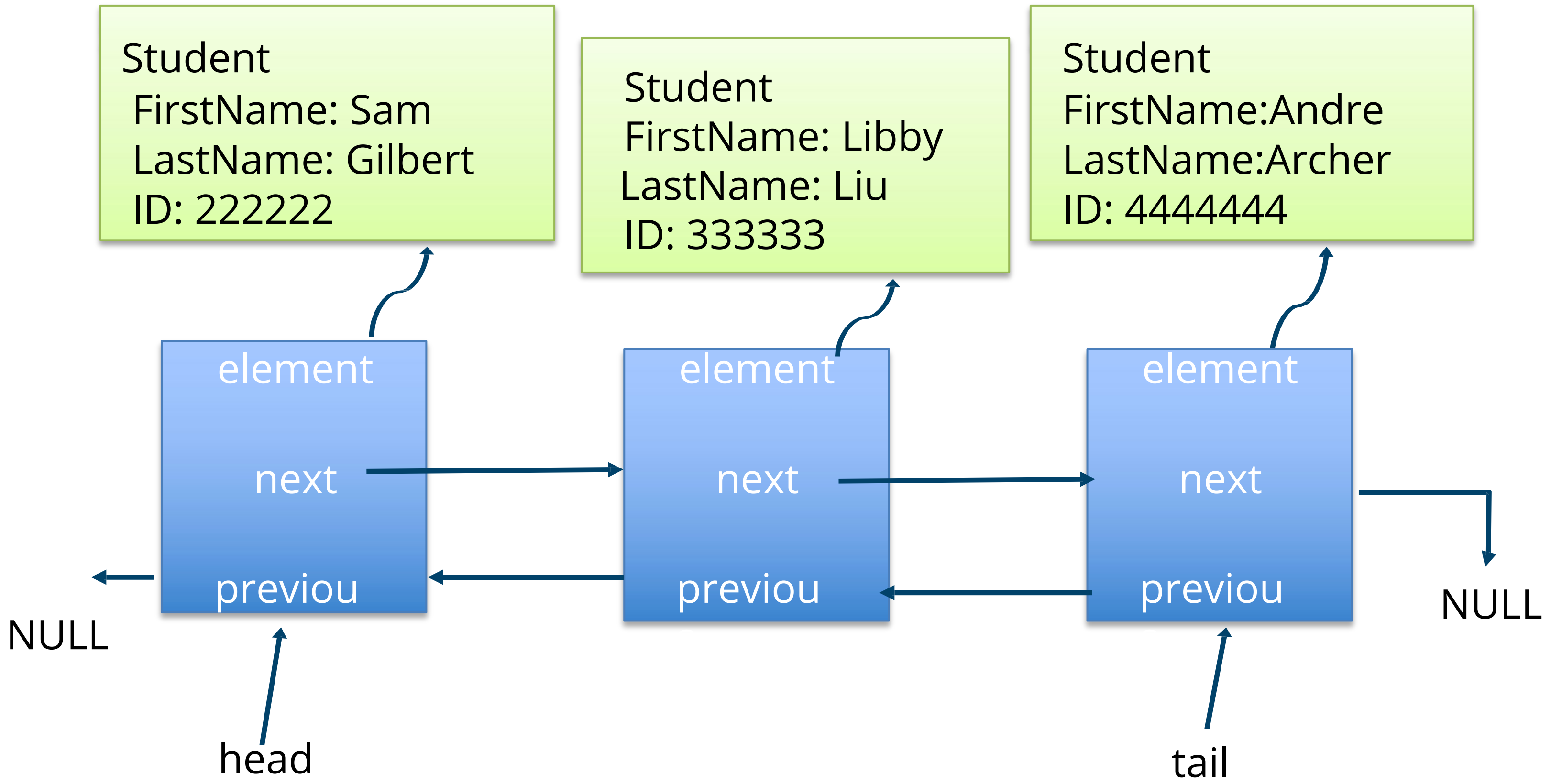
- `boolean enqueue(T element)` O(1)
  - Appends the specified element to the end of this list.
- `T dequeue()` O(1)
  - Removes the first element from this list, if it is present.
- `T first()`
  - Returns but does not remove the first element from this list, if it is present.
- `boolean isEmpty()`
  - returns true if the queue is empty.
- `int size()`
  - Returns the number of elements in the queue
- `String toString()`
  - string representation of the queue

At worst, how long do these take?

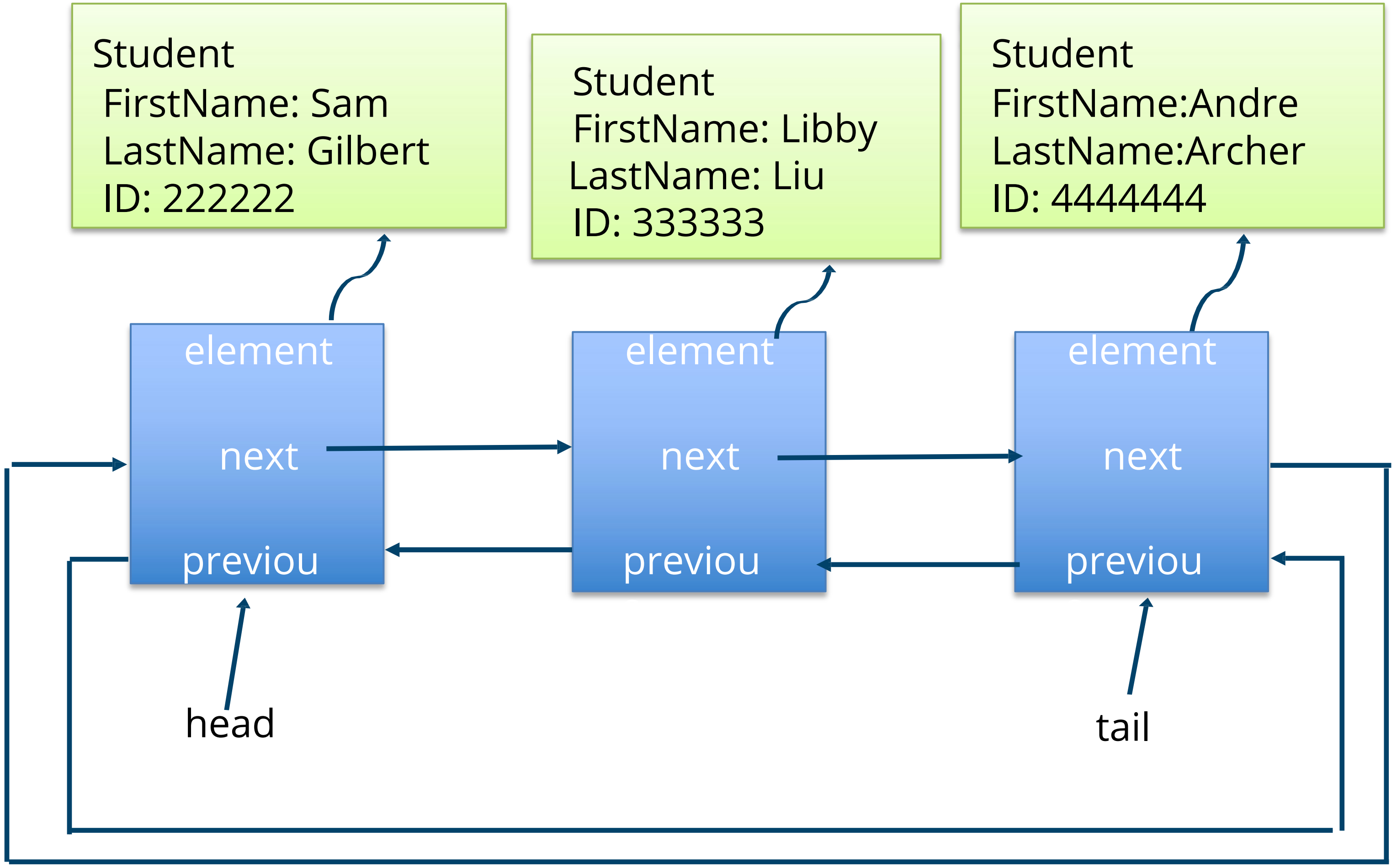
- `boolean enqueue(T element)` O(1)
    - Appends the specified element to the end of this list.
- `T dequeue()` O(1)
    - Removes the first element from this list, if it is present.
- `T first()` O(1)
    - Returns but does not remove the first element from this list, if it is present.
- `boolean isEmpty()` O(1)
    - returns true if the queue is empty.
- `int size()` O(1)
    - Returns the number of elements in the queue
- `String toString()`
    - string representation of the queue

At worst, how long do these take?

- `boolean enqueue(T element)` O(1)
  - Appends the specified element to the end of this list.
- `T dequeue()` O(1)
  - Removes the first element from this list, if it is present.
- `T first()` O(1)
  - Returns but does not remove the first element from this list, if it is present.
- `boolean isEmpty()` O(1)
  - returns true if the queue is empty.
- `int size()` O(1)
  - Returns the number of elements in the queue
- `String toString()` O(n)
  - string representation of the queue



Linked List data structures can be single-linked (only a next pointer) or double-linked (next and previous) which allows you to iterate in both directions.



They can also be circular.

In-class Activity

Maze Activity (day one)