# CSIS 3103

## Linked List Implementations

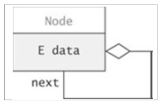
## Linked Lists

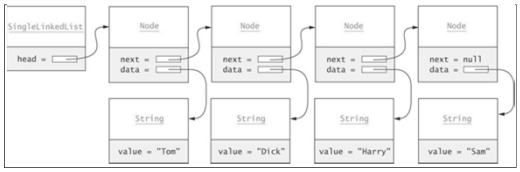
`ArrayList` `add` and `remove` methods are O(n)

Linked lists can add or remove in O(1) in some cases

## Single-linked List Nodes

A node contains a data item and reference to a node (link)



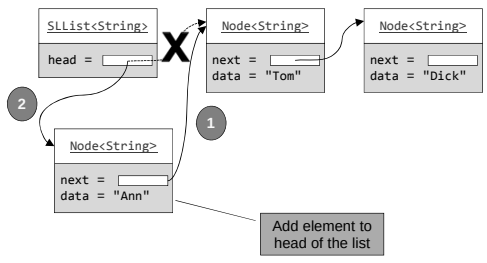

```
Node<String> head = new Node<String>("Tom");
Node<String> head.next = new Node<String>("Dick");
Node<String> head.next.next = new Node<String>("Harry");
Node<String> head.next.next.next = new Node<String>("Sam");
```

## `SingleLinkedList` Class

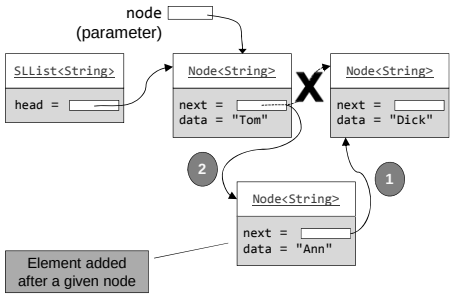
| Method | Behavior |
|---|---|
| `public E get(int index)` | Returns a reference to the element at position `index`. |
| `public E set(int index, E anEntry)` | Sets the element at position `index` to reference `anEntry`. Returns the previous value. |
| `public int size()` | Gets the current size of the `List`. |
| `public boolean add(E anEntry)` | Adds a reference to `anEntry` at the end of the `List`. Always returns `true`. |
| `public void add(int index, E anEntry)` | Adds a reference to `anEntry`, inserting it before the item at position `index`. |
| `int indexOf(E target)` | Searches for `target` and returns the position of the first occurrence, or –1 if it is not in the `List`. |

Several helper methods are used to implement these operations

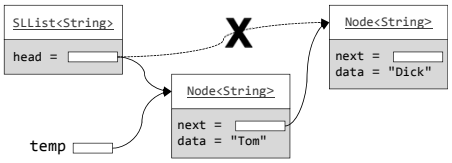
## `addFirst(E)`



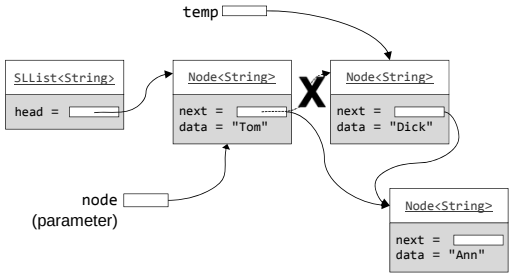

## `addAfter(Node<E>, E)`



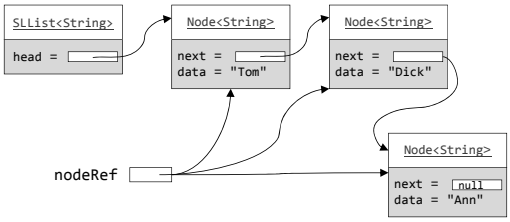
## removeFirst()




## removeAfter(Node<E>)




## Traversing a Single-Linked List

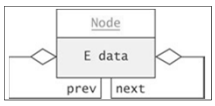



## Characteristics of single-linked list

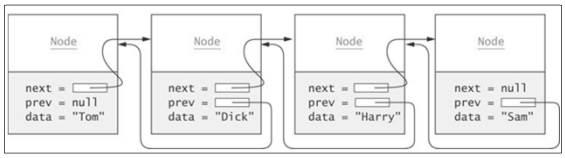
- Insert/remove at the front of the list is O(1)
- Insert/remove at other positions is O(*n*)
- Insert/remove require a reference to the previous node
- Can traverse the list only in the forward direction

The double-linked list overcomes some of these limitations

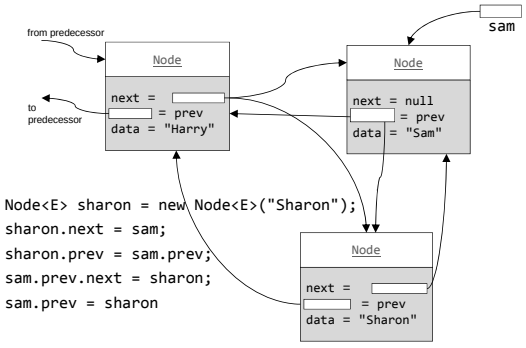
## Double-Linked Lists



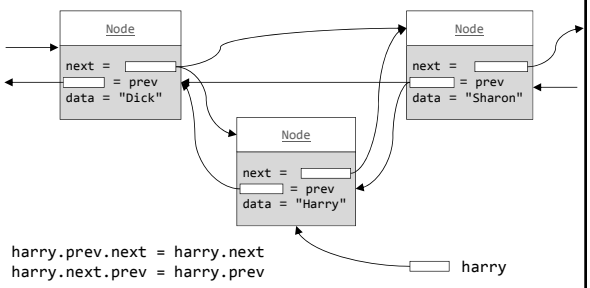

A node contains a data item and references to previous and next nodes




## Inserting in a Double-Linked List



## Removing from a Double-Linked List




## A Double-Linked Class

A double-linked list object has data fields:

- `head` (reference to first list `Node`)
- `tail` (reference to last list `Node`)
- `size`


Insertion at either end is $O(1)$

- And require special cases

Insertion elsewhere is still $O(n)$

## Circular Lists

Singly-linked circular lists

- Link last node to the first node

Circular double-linked list

- Link last node to the first node
- Link first node to the last node

**Advantages:**

- Continue to traverse even after passing the first or last node
- Visit all elements from any starting point
- Never fall off the end of a list

**Disadvantage**: Code must avoid an infinite loop!

## Circular Lists