Week 11 - Wednesday

COMP 2000

Last time

- What did we talk about last time?
- Exam 2
- Before that:
  - Review
- Before that:
  - Networking

Questions?

Project 3

Dynamic Data Structures

Storing a bunch of stuff

- What if you want to hold a lot of **`int`** values, or **`String`** values, or **`Wombat`** values?
- You make an array!
  - But arrays have a fixed size
- What if you don't know how long to make it?
  - You have to overestimate how many values you need
  - Or you have to periodically resize your array

Dynamic data structures

- Another approach is using a **dynamic data structure**
- A dynamic data structure grows as you need space
- Python has lists (and sets and dictionaries) built in
- But Java depends on libraries
- Before we do libraries, let's implement a linked list ourselves to see what a pain it is
- Making data structures that work efficiently in different circumstances is the heart of COMP 2100

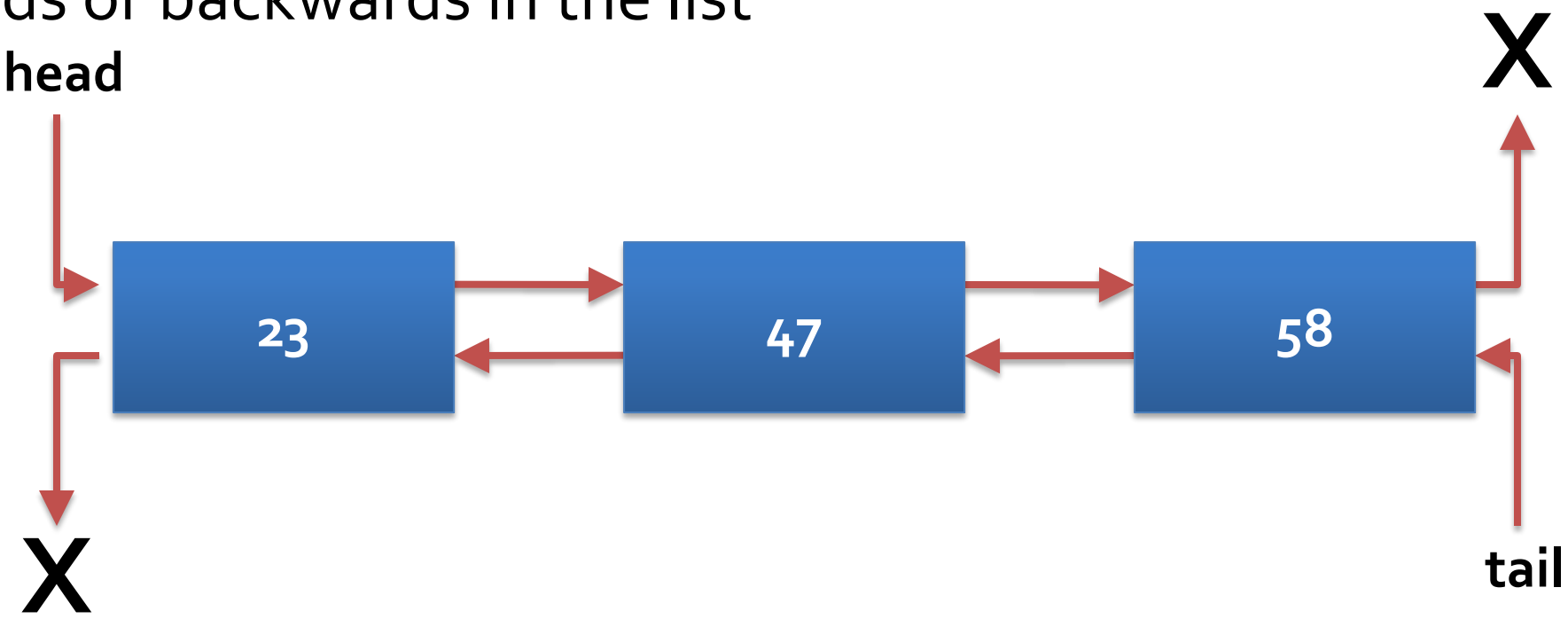
Linked Lists

Linked list

- A linked list is one of the simplest kinds of dynamic data structures
- You can imagine a linked list as a train
- Each node in the linked list has some cargo, and it can point at the next item in the list
- The last item points at null so that you know that the train has ended
- You can add and remove nodes as much as you want, and nothing needs to be resized

Doubly linked list

- The most common library implementation of a linked list is a **doubly linked list**
- Node consists of data, a next pointer, and a previous pointer
- Because we know the next and the previous, we can move forwards or backwards in the list



Definition

- Let's try a simple definition for a doubly linked list that holds an unlimited number of **`String`** values:

```
public class LinkedList {
   private static class Node {
        public String data;
        public Node next;
        public Node previous;
   }

   private Node head = null
   private Node tail = null;
   private int size = 0;
   …
}
```

Linked list methods

- Inside the **`LinkedList`** class, we have to write methods to manipulate it
- There will be simple accessor methods like **`size()`** that return the size
- There will be simple mutator methods like **`clear()`** that remove all the elements from the list
- But the hard work will be methods to get, add, remove, and find elements

Easy methods

- If we always keep the size member correctly updated, the **`size()`** accessor has a straightforward implementation

```
public int size() {
      return size;
}
```

- Likewise, clearing the list returns it to its state right after construction

```
public void clear() {
   head = null;
   tail = null;
   size = 0;
}
```

Add to the end of the list

- Method signature:

```
public void add(String value)
```

- The method creates a new node
- If the list is empty, it points **`head`** at the new node
- Otherwise, it points the **`tail`** node's **`next`** at the new node and the new node's previous at the **`tail`** node
- It updates the **`tail`** to point at the new node
- It increases **`size`** by one

Get an element from the list

- Method signature:

```
public String get(int index)
```

- If **`index`** is illegal, throw an **`IndexOutOfBoundsException`**
- Loop through the list until reaching the node at location **`index`** (using 0-based indexing, because we are computer scientists!)
- Return the **`data`** of the node in question

Remove the first element

- Method signature:

```
public String remove()
```

- If the list is empty, throw a **`NoSuchElementException`**
- Point a temporary variable at the **`head`** node
- Point **`head`** at the next node
- If the next node is null, point **`tail`** at **`null`**
- Otherwise, point the next node's **`previous`** at **`null`**
- Return the **`data`** of the temporary node

Find the index of an element

- Method signature:

```
public int indexOf(String value)
```

- Loop through the list until reaching a node whose **`data`** is equal to **`value`**, keeping a counter of the current index
- If **`value`** is found, return the index
- If **`value`** is never found, return **`-1`**

Upcoming

Next time…

- Generics

Reminders

- **Finish Project 3**
  - **Due Friday by midnight!**
- Read Chapter 18